

LAPORAN UJIAN AKHIR SEMESTER
“IMPLEMENTASI KONSEP OOP DALAM PEMBUATAN PROGRAM KLINIK
SEDERHANA MENGGUNAKAN JAVA”
UNTUK MEMENUHI TUGAS MATA KULIAH PEMROGRAMAN BERBASIS OBJEK



Disusun Oleh :
Kelompok 2

1. Azzelya Rosya Denovya 2310631250041
2. Mochamad Ridwan Al Anshori 2310631250062
3. Tito Adi Prasetyo 2310631250035
4. Nazwa Chandra Febriyanti 2310631250025

Kelas: 4B Sistem Informasi
Dosen Pengampu: Taufik Ridwan S.T, M.T.

PROGAM STUDI SISTEM INFORMASI
FAKULTAS ILMU KOMPUTER
UNIVERSITAS SINGAPERBANGSA KARAWANG 2024/2025

KATA PENGANTAR

Puji syukur kami panjatkan ke hadirat Allah SWT atas rahmat dan karunia-Nya sehingga kami dapat menyelesaikan laporan Ujian Akhir Semester dengan judul “Implementasi Konsep OOP dalam Pembuatan Program Klinik Sederhana Menggunakan Java” tepat waktu.

Laporan ini disusun untuk memenuhi tugas Mata Kuliah Pemrograman Berorientasi Objek, dengan fokus pada pengembangan aplikasi klinik yang mencakup fitur seperti pendaftaran pasien, manajemen dokter, jadwal praktik, resep obat, dan pengelolaan ruangan serta obat. Seluruh fitur dikembangkan menggunakan konsep Object Oriented Programming dalam bahasa Java.

Kami mengucapkan terima kasih kepada Bapak Taufik Ridwan, S.T., M.T. selaku dosen pengampu, serta semua pihak yang telah memberikan dukungan. Semoga laporan ini bermanfaat dan dapat menjadi referensi bagi pengembangan aplikasi serupa di masa depan.

Karawang, 26 Mei 2025

Kelompok 2

Daftar Isi

KATA PENGANTAR.....	2
Daftar Isi.....	3
BAB I.....	4
PENDAHULUAN	4
1.1 Latar Belakang	4
1.2 Rumusan Masalah.....	5
1.3 Tujuan Masalah	5
BAB II	7
LANDASAN TEORI	7
2.1 Konsep Dasar Object-Oriented Programming (OOP) Programming (OOP).....	7
2.2 Fungsi Utama Perangkat Lunak	7
BAB III.....	9
PEMBAHASAN	9
3.1 Perancangan Sistem (UML).....	9
3.1.1 Class Diagram.....	9
3.1.2 Use Case Diagram.....	10
3.1.3 Activity Diagram	11
3.1.4 Sequence Diagram	15
3.2 Implementasi	23
3.2.1 Folder Controller.....	23
3.2.2 Folder Data Access Object (DAO)	36
3.2.3 Folder Klinik PBO	52
3.2.4 Folder Model.....	104
BAB IV	114
KESIMPULAN	114

BAB I

PENDAHULUAN

1.1 Latar Belakang

Dalam era digital yang semakin berkembang pesat, pemanfaatan program berbasis GUI (Graphical User Interface) menjadi kebutuhan penting di berbagai sektor, termasuk dalam bidang layanan kesehatan seperti klinik. Klinik modern membutuhkan sistem yang efisien, mudah digunakan, dan berbasis GUI untuk mengelola berbagai aspek operasional, mulai dari pendaftaran pasien, manajemen dokter, jadwal praktik, hingga pengelolaan obat dan ruangan.

Pengembangan program klinik sederhana dengan pendekatan Object-Oriented Programming (OOP) menggunakan bahasa pemrograman Java menjadi solusi yang tepat. Konsep OOP memungkinkan pengembang untuk memodelkan sistem secara modular, dengan memisahkan data dan fungsionalitas dalam bentuk kelas-kelas, sehingga memudahkan pengelolaan dan pengembangan aplikasi dalam jangka panjang. Java sebagai bahasa pemrograman yang stabil dan multiplatform juga sangat mendukung pembuatan program GUI yang handal dan mudah dioperasikan.

Program klinik sederhana yang dikembangkan dalam proyek ini dirancang untuk mencakup beberapa fitur utama, seperti pendaftaran pasien, pengelolaan data dokter, penjadwalan praktik, pembuatan resep obat, serta pengaturan ruangan dan stok obat. Setiap komponen tersebut diimplementasikan dalam bentuk objek, yang saling berinteraksi sesuai prinsip OOP. Selain itu, integrasi fitur CRUD (Create, Read, Update, Delete) juga diterapkan untuk memudahkan admin dan pengguna dalam mengelola data klinik secara menyeluruh.

Melalui laporan ini, akan dibahas secara detail bagaimana konsep OOP diimplementasikan dalam pengembangan program klinik sederhana menggunakan Java. Diharapkan, program ini dapat menjadi solusi efektif dalam membantu proses operasional di klinik serta menjadi media pembelajaran yang aplikatif dalam memahami pemrograman berorientasi objek.

1.2 Rumusan Masalah

Dalam rumusan masalah ini, akan dibahas secara mendalam beberapa aspek yang berkaitan dengan pengembangan program klinik sederhana. Adapun aspek-aspek yang akan dibahas meliputi:

- Bagaimana implementasi konsep Object-Oriented Programming (OOP) dalam pengembangan program klinik sederhana menggunakan bahasa pemrograman Java?
- Bagaimana efektivitas dan efisiensi penggunaan program klinik sederhana dalam membantu proses administrasi dan pelayanan klinik?
- Bagaimana proses pendaftaran pasien dilakukan oleh admin, dan bagaimana data pasien tercatat dan tersimpan dalam sistem?
- Bagaimana sistem mendaftarkan dan menampilkan jadwal praktik, jumlah pasien, jumlah dokter, dan jumlah ruangan.

Dengan rumusan masalah ini, diharapkan dapat memberikan gambaran yang jelas mengenai penerapan konsep OOP dalam pengembangan program klinik, serta bagaimana program ini dapat meningkatkan efisiensi dan efektivitas dalam operasional pelayanan kesehatan di lingkungan klinik.

1.3 Tujuan Masalah

Dalam laporan ini, akan diuraikan secara rinci tujuan-tujuan yang ingin dicapai dari implementasi program klinik sederhana. Tujuan tersebut antara lain:

- Menerapkan konsep Object-Oriented Programming (OOP) dalam pengembangan program klinik sederhana menggunakan bahasa pemrograman Java.
- Meningkatkan efektivitas dan efisiensi pengelolaan layanan dan administrasi di klinik melalui sistem informasi yang terintegrasi.
- Mempermudah admin dalam proses pendaftaran pasien dan pengelolaan data, seperti menambahkan, melihat, mengubah, dan menghapus informasi pasien.
- Menyediakan fitur pengaturan dan tampilan jadwal praktik dokter, jumlah pasien, jumlah dokter, dan jumlah ruangan, serta pengelolaan informasi ruangan klinik secara sistematis.

Dengan tujuan-tujuan ini, diharapkan program klinik sederhana dapat menjadi solusi yang efektif dalam pengelolaan operasional klinik dan mendukung peningkatan kualitas pelayanan kesehatan.

BAB II

LANDASAN TEORI

2.1 Konsep Dasar Object-Oriented Programming (OOP)

Object-Oriented Programming (OOP) adalah paradigma pemrograman yang berfokus pada pembuatan objek yang mengandung data (atribut) dan perilaku (metode). Konsep utama OOP meliputi:

1. Kelas (Class): Blueprint untuk membuat objek. Contoh: Pasien, Dokter, Obat.
2. Objek (Object): Instansi dari kelas yang memiliki data dan metode. Contoh: pasien1 dengan nama "Azzelya".
3. Enkapsulasi (Encapsulation): Membungkus data dan metode dalam kelas, serta mengontrol akses melalui modifier (public, private, protected).
4. Inheritance (Pewarisan): Kelas anak mewarisi sifat kelas parent. Contoh: Dokter bisa mewarisi Karyawan.
5. Polimorfisme (Polymorphism): Kemampuan objek untuk memiliki banyak bentuk. Contoh: metode tampilkanInfo() di kelas Pasien dan Dokter dengan implementasi berbeda.
6. Abstraksi (Abstraction): Menyembunyikan detail kompleks dan menampilkan fungsionalitas esensial.

Java dipilih karena mendukung penuh OOP, multiplatform, dan memiliki library GUI seperti Swing/JavaFX untuk antarmuka pengguna.

2.2 Fungsi Utama Perangkat Lunak

Berikut adalah penjelasan mengenai fitur utama dari perangkat lunak yang dibuat untuk Program Klinik Sederhana:

1. Pendaftaran Pasien
Fitur ini memungkinkan admin untuk mendaftarkan pasien baru ke sistem. Data yang dimasukkan bisa berupa nama, umur, alamat, nomor kontak, dan keluhan awal.
2. List Obat
Menampilkan daftar semua obat yang tersedia di klinik, termasuk nama obat, jenis, dan stok.
3. List Dokter
Menampilkan informasi dokter yang bekerja di klinik, seperti nama, spesialisasi, dan jadwal praktik.

4. List User

Menampilkan user yang ada di klinik dan perannya

5. Jadwal Praktik

Menyimpan dan menampilkan jadwal praktik dokter. Dapat digunakan oleh admin untuk pengaturan, dan oleh dokter untuk melihat jadwal dirinya.

6. Resep Obat

Dokter dapat membuat resep obat berdasarkan data pasien. Resep berisi nama pasien, obat yang diresepkan, dan dosisnya.

7. List Ruangan

Menampilkan daftar ruangan yang ada di klinik, seperti ruang periksa, ruang tunggu, ruang perawatan, dll.

Dengan adanya fitur-fitur tersebut, penggunaan program klinik sederhana dapat membantu meningkatkan efektivitas dan efisiensi pelayanan kesehatan. Admin dapat dengan mudah mendaftarkan pasien baru dan mengelola informasi penting seperti data dokter, jadwal praktik, serta ketersediaan obat dan ruangan. Dokter pun terbantu dengan fitur resep obat yang memungkinkan penulisan resep secara sistematis dan terdokumentasi. Selain itu, pasien dapat dilayani dengan lebih cepat karena semua informasi tersedia secara terstruktur dan terpusat. Secara keseluruhan, aplikasi ini mampu mengurangi kesalahan pencatatan manual dan mendukung kelancaran operasional klinik sehari-hari.

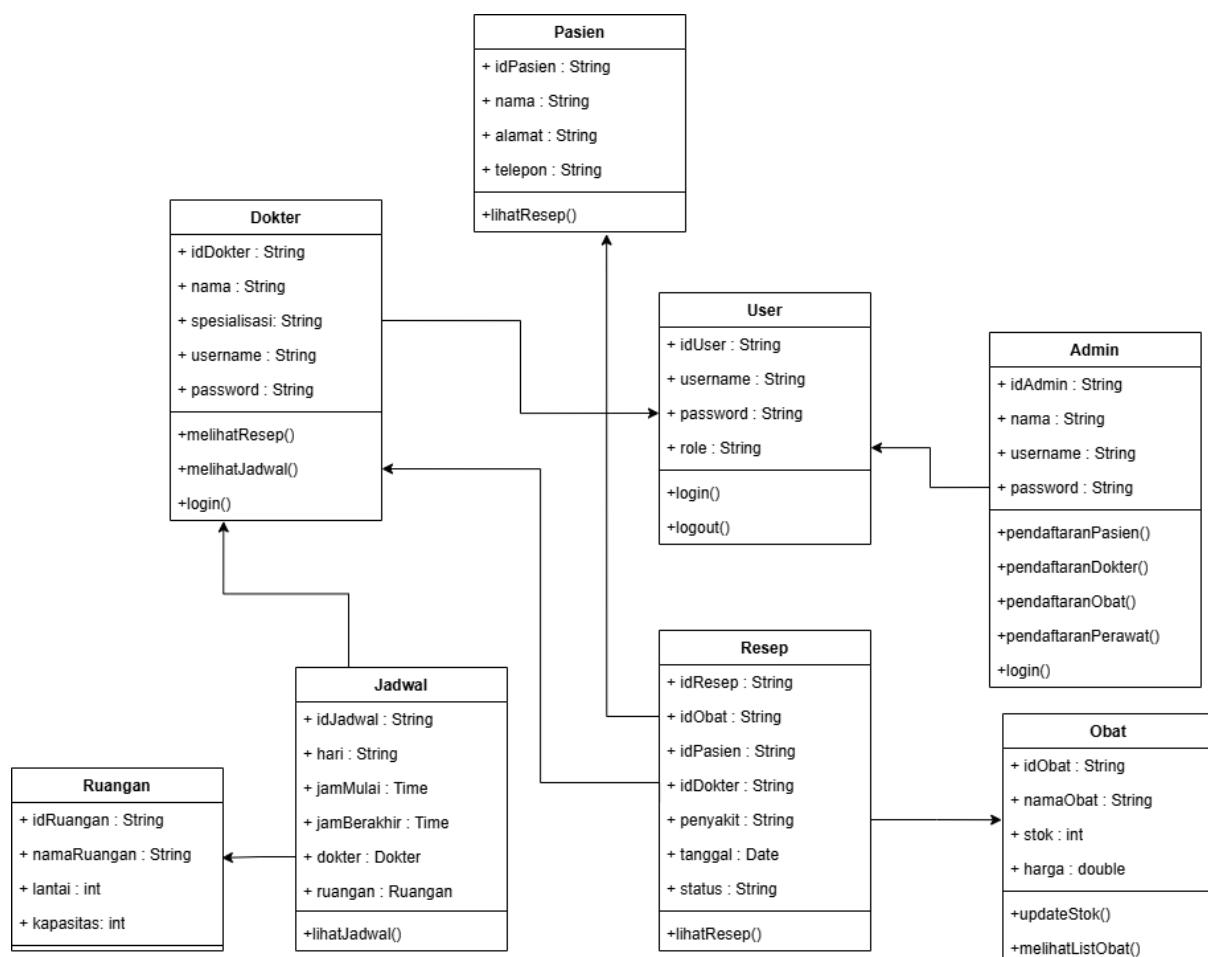
BAB III

PEMBAHASAN

3.1 Perancangan Sistem (UML)

Dalam konteks perancangan program klinik sederhana, diagram pada Unified Modeling Language (UML) digunakan untuk memvisualisasikan struktur dan perilaku sistem secara menyeluruh. Melalui berbagai jenis diagram seperti class diagram, use case, activity diagram, sequence diagram, dan interaksi aktor dengan sistem, perancangan menjadi lebih terarah dan terdokumentasi dengan baik.

3.1.1 Class Diagram



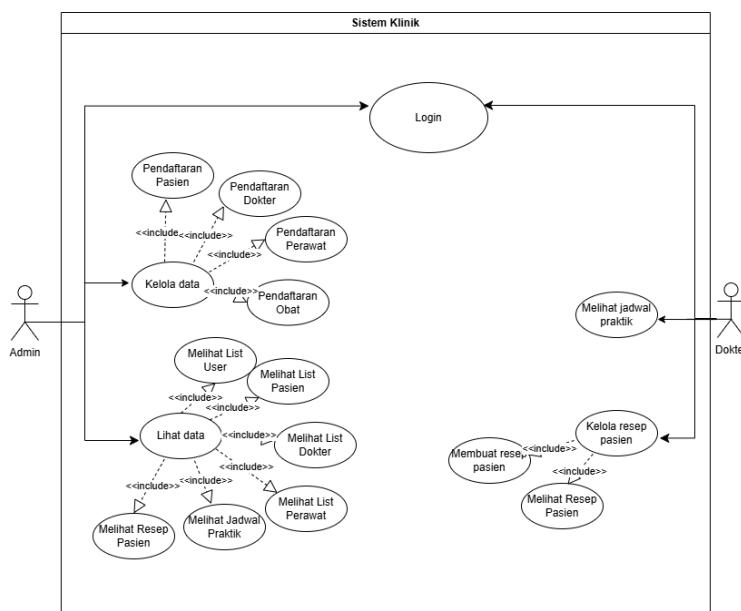
Class diagram ini menggambarkan struktur utama dari sistem informasi klinik sederhana, yang terdiri dari beberapa kelas beserta atribut dan metode yang dimilikinya. Setiap kelas mewakili entitas yang memiliki peran penting dalam sistem.

- **Pasien** menyimpan data pasien seperti id, nama, alamat, dan nomor telepon. Pasien juga memiliki fungsi untuk melihat resep.
- **Dokter** memiliki atribut seperti id, nama, spesialisasi, username, dan password. Dokter dapat melakukan login serta melihat resep dan jadwal praktiknya.

- **User** adalah kelas umum yang menyimpan informasi akun pengguna seperti id, username, password, dan peran (*role*). Tersedia juga metode login dan logout.
- **Admin** merupakan bagian dari user yang memiliki peran khusus, yaitu dapat melakukan pendaftaran data seperti pasien, dokter, obat, dan perawat. Admin juga memiliki metode login sendiri.
- **Obat** berisi informasi terkait obat di klinik, seperti id, nama, stok, dan harga. Obat dapat diperbarui stoknya dan ditampilkan dalam daftar.
- **Resep** menyimpan informasi resep yang berisi id resep, id pasien, id dokter, id obat, penyakit, tanggal, dan status resep. Pasien dan dokter dapat melihat isi resep ini.
- **Jadwal** digunakan untuk mencatat jadwal praktik dokter, yang terdiri dari hari, jam mulai, jam berakhir, serta informasi dokter dan ruangan yang digunakan.
- **Ruangan** menyimpan informasi mengenai ruang-ruang di klinik, seperti nama ruangan, lantai, dan kapasitas.

Class diagram ini menunjukkan bagaimana masing-masing entitas saling berhubungan dan berinteraksi dalam sistem. Hubungan antar kelas juga mendukung proses bisnis utama klinik, seperti pendaftaran pasien, pemberian resep, pengaturan jadwal dokter, hingga pengelolaan data obat dan ruangan.

3.1.2 Use Case Diagram



Use case diagram ini menggambarkan interaksi antara aktor (pengguna) dan sistem klinik, yang mencakup dua aktor utama, yaitu Admin dan Dokter. Diagram ini menunjukkan fungsi-fungsi utama yang dapat dilakukan oleh masing-masing aktor.

1. Aktor:

- Admin: Memiliki akses penuh untuk mengelola dan melihat data di sistem.

- Dokter: Berperan dalam melihat jadwal praktik dan mengelola resep pasien.

2. Fitur yang Diakses Admin:

- Login: Admin harus masuk ke sistem terlebih dahulu.
- Kelola Data: Admin dapat mengakses menu kelola data, yang mencakup: Pendaftaran Pasien, Pendaftaran Dokter, Pendaftaran Perawat, Pendaftaran Obat.
- Lihat Data: Admin juga dapat melihat berbagai daftar informasi seperti: List User, List Pasien, List Dokter, List Perawat, Jadwal Praktik, dan Resep Pasien.

Relasi <<include>> menunjukkan bahwa fungsi-fungsi seperti pendaftaran dan melihat daftar merupakan bagian dari proses utama *Kelola Data* dan *Lihat Data*.

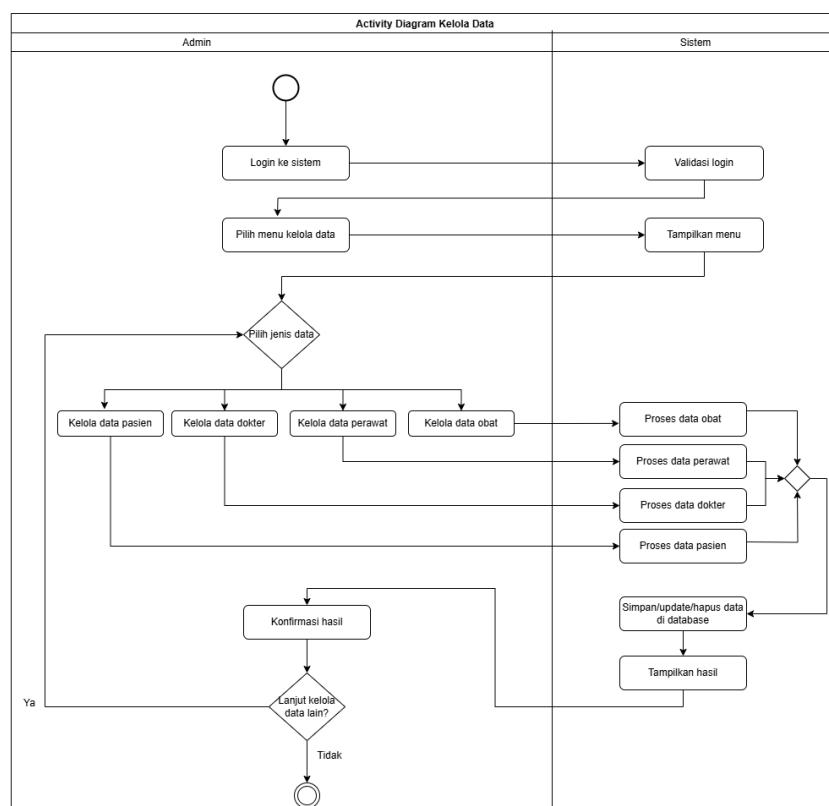
3. Fitur yang Diakses Dokter:

- Login: Dokter juga harus login terlebih dahulu.
- Melihat Jadwal Praktik: Dokter dapat melihat jadwal tugas/praktik mereka.
- Kelola Resep Pasien: Dokter dapat membuat resep dan melihat resep pasien yang pernah dibuat.

Dalam *Kelola Resep Pasien*, terdapat dua aktivitas yang termasuk di dalamnya, yaitu *Membuat Resep Pasien* dan *Melihat Resep Pasien*.

3.1.3 Activity Diagram

a. Activity Diagram Kelola Data

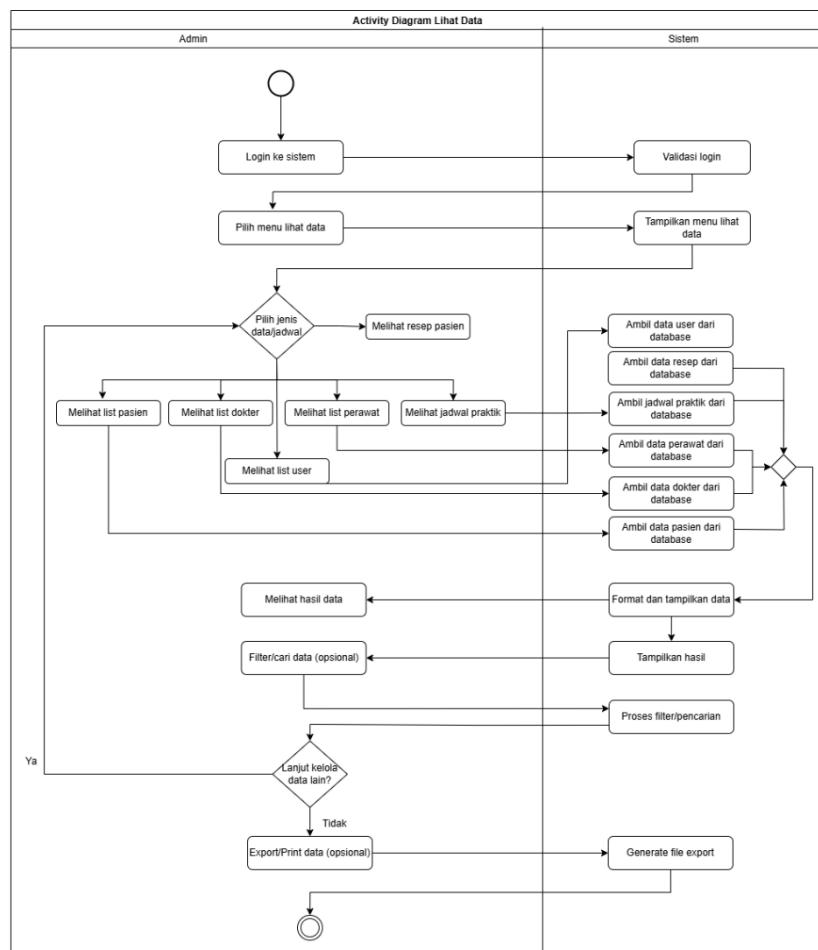


Proses dimulai saat Admin login ke sistem, kemudian sistem melakukan validasi login. Setelah berhasil login, Admin memilih menu kelola data, lalu sistem menampilkan menu yang tersedia. Admin dapat memilih salah satu dari beberapa jenis data untuk dikelola:

- Kelola data pasien → sistem memproses data pasien sesuai aksi (tambah, ubah, hapus).
- Kelola data dokter → sistem memproses data dokter sesuai aksi.
- Kelola data perawat → sistem memproses data perawat sesuai aksi.
- Kelola data obat → sistem memproses data obat sesuai aksi.

Setelah data diproses, sistem akan menyimpan/update/hapus data di database dan menampilkan hasilnya. Admin diminta untuk melakukan konfirmasi hasil, lalu memutuskan apakah ingin melanjutkan ke pengelolaan data lain atau mengakhiri proses. Semua aktivitas yang berhasil akan diakhiri dengan notifikasi hasil dari sistem.

b. Activity Diagram Lihat Data



Proses dimulai saat Admin login ke sistem, kemudian sistem melakukan validasi login. Setelah berhasil login, Admin memilih menu lihat data, lalu sistem menampilkan pilihan menu

lihat data. Admin kemudian diminta untuk memilih jenis data atau jadwal yang ingin dilihat. Ada beberapa pilihan jenis data/jadwal yang tersedia:

- Melihat resep pasien → sistem mengambil data resep dari database.
- Melihat list pasien → sistem mengambil data pasien dari database.
- Melihat list dokter → sistem mengambil data dokter dari database.
- Melihat list user → sistem mengambil data user dari database.
- Melihat list perawat → sistem mengambil data perawat dari database.
- Melihat jadwal praktik → sistem mengambil data jadwal praktik dari database.

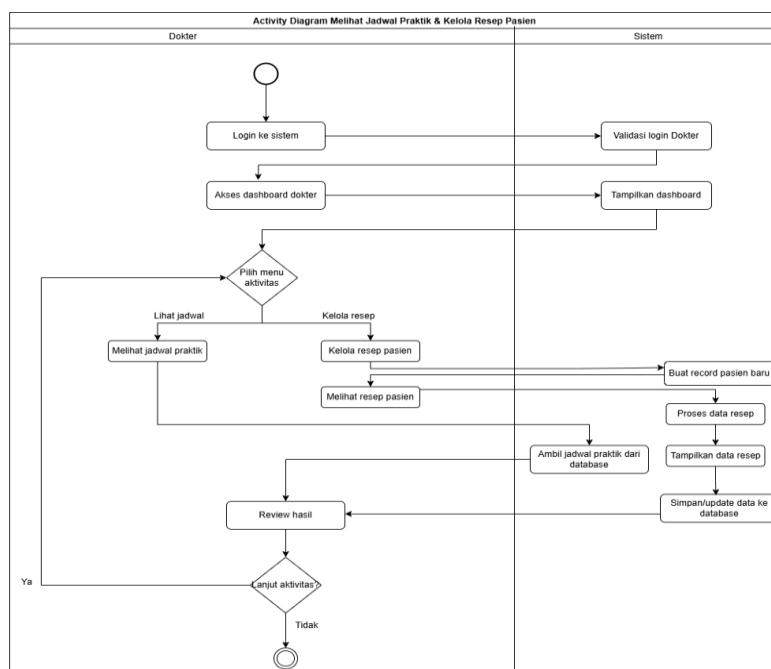
Setelah data yang diminta diambil, sistem akan melakukan format dan menampilkan data tersebut kepada Admin.

Admin kemudian dapat:

- Melihat hasil data, dan
- (Opsional) Melakukan filter atau pencarian data tertentu, yang akan diproses dan ditampilkan oleh sistem.

Setelah itu, Admin memutuskan apakah ingin melanjutkan untuk mengelola data lain. Jika ya, maka proses akan kembali ke pemilihan jenis data/jadwal. Jika tidak, maka Admin dapat mengekspor atau mencetak data (opsional), dan proses pun berakhir.

c. Activity Diagram Melihat Jadwal Praktik & Kelola Resep Pasien



Proses dimulai saat Dokter login ke sistem, kemudian sistem melakukan validasi login Dokter. Setelah berhasil login, Dokter mengakses dashboard, lalu sistem menampilkan tampilan dashboard. Selanjutnya, Dokter diminta untuk memilih menu aktivitas. Terdapat dua pilihan utama:

1. Lihat jadwal praktik

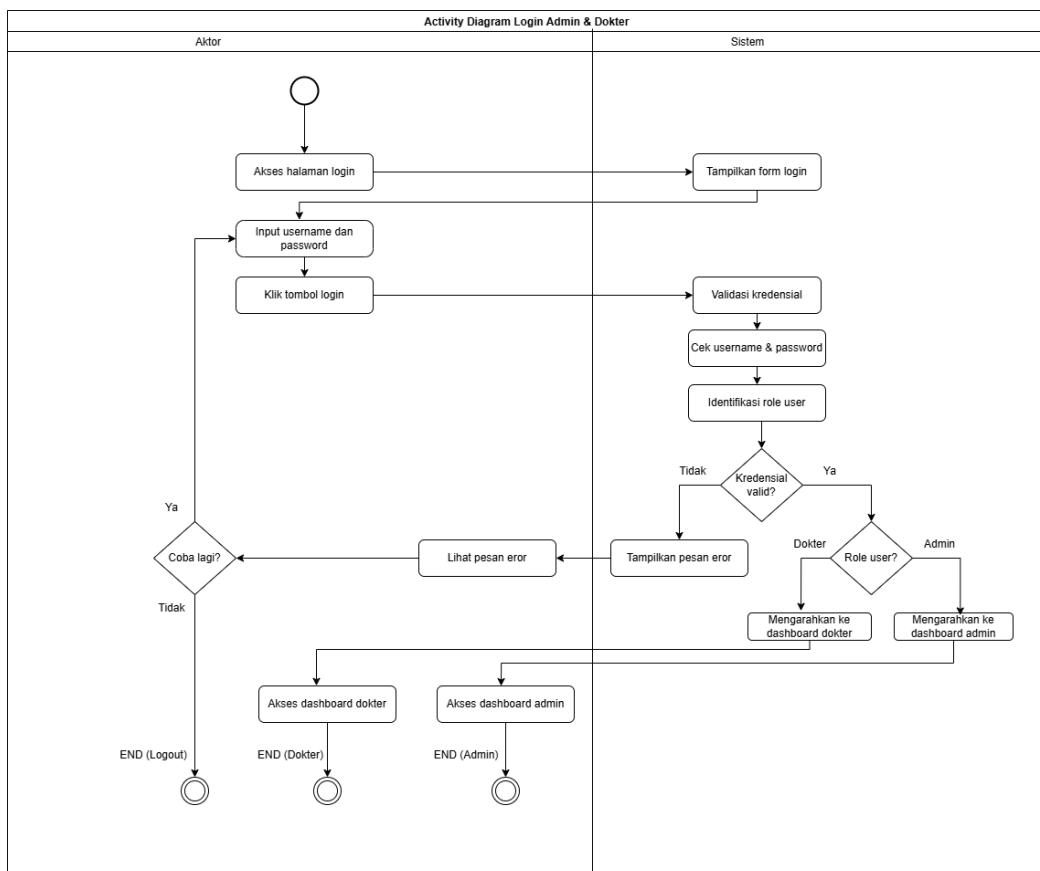
Dokter memilih untuk melihat jadwal praktik, kemudian sistem mengambil data jadwal praktik dari database, dan menampilkannya kepada dokter.

2. Kelola resep pasien

Dokter memilih menu kelola resep, lalu Dokter dapat melihat resep pasien atau mengelola resep pasien. Sistem akan memeriksa apakah perlu membuat record pasien baru (jika belum ada), kemudian memproses data resep yang diberikan dokter, Lalu menampilkan data resep, da menyimpan atau mengupdate data ke database.

Setelah aktivitas selesai, Dokter akan melakukan review terhadap hasil aktivitas. Kemudian sistem menanyakan apakah dokter ingin melanjutkan aktivitas lain. Jika ya, maka proses kembali ke pemilihan menu aktivitas. Jika tidak, maka proses berakhir.

d. Activity Diagram Login Admin & Dokter



Proses dimulai saat aktor (baik Admin maupun Dokter) mengakses halaman login, lalu sistem menampilkan form login. Pengguna kemudian memasukkan username dan password, lalu menekan tombol login. Setelah itu, sistem melakukan validasi kredensial dengan:

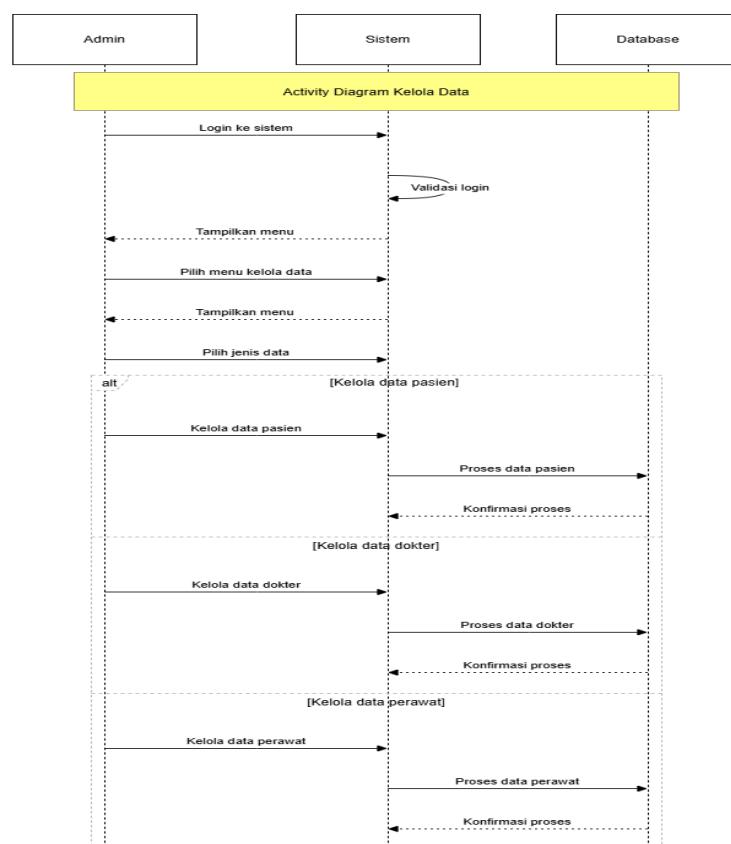
1. Memeriksa kecocokan username dan password.
2. Mengidentifikasi peran (role) dari user.

Selanjutnya, sistem memeriksa apakah kredensial valid:

- Jika tidak valid, sistem akan menampilkan pesan error, dan pengguna akan diarahkan untuk memutuskan apakah ingin mencoba login lagi:
 - Jika ya, kembali ke proses input login.
 - Jika tidak, maka proses login diakhiri (logout).
- Jika valid, sistem akan mengecek role user:
 - Jika role-nya dokter, maka sistem akan mengarahkan ke dashboard dokter, dan proses login berakhir (END Dokter).
 - Jika role-nya admin, maka sistem akan mengarahkan ke dashboard admin, dan proses login berakhir (END Admin).

3.1.4 Sequence Diagram

a. sequence diagram kelola data



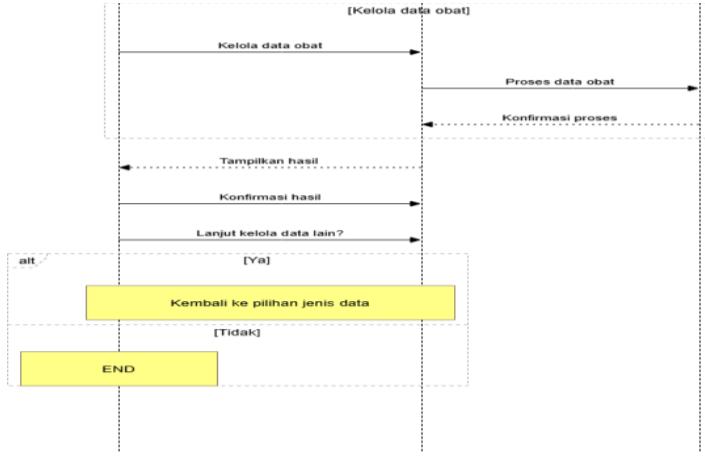
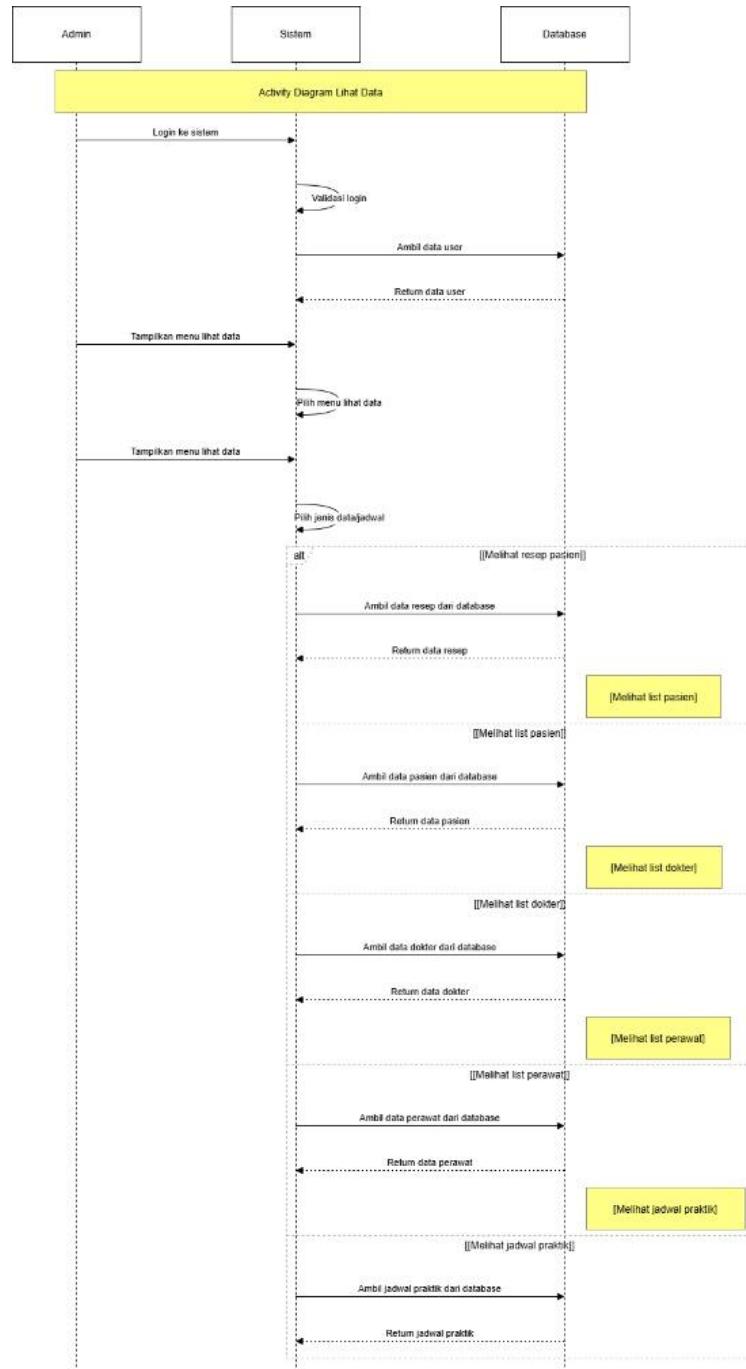


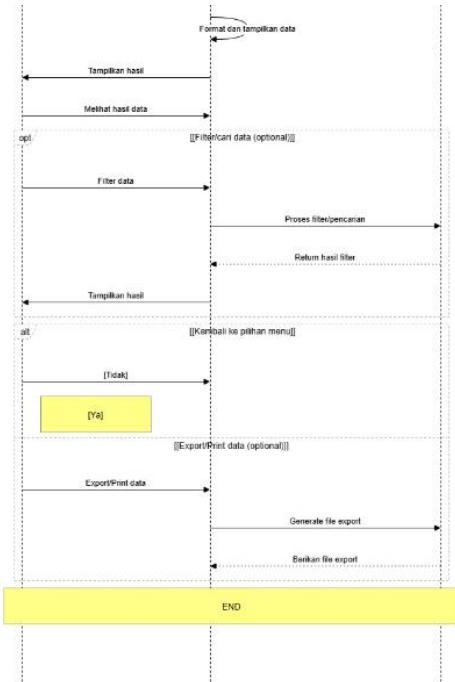
Diagram ini menunjukkan fungsi admin dalam mengelola berbagai jenis data dalam sistem dengan fokus pada empat kategori data utama:

- Fitur Utama Admin:
 1. Login dan Validasi: Admin login ke sistem dengan validasi yang berhasil
 2. Menu Kelola Data: Admin memilih menu kelola data dari dashboard utama
- Jenis Data yang Dikelola:
 1. Kelola Data Pasien
 - Admin memilih opsi "Kelola data pasien"
 - Sistem memproses data pasien ke database
 - Database memberikan konfirmasi proses berhasil
 2. Kelola Data Dokter
 - Admin memilih opsi "Kelola data dokter"
 - Sistem melakukan pemrosesan informasi dokter
 - Database mengkonfirmasi penyimpanan data dokter
 3. Kelola Data Perawat
 - Admin memilih opsi "Kelola data perawat"
 - Sistem mengelola dan memproses data perawat
 - Database memberikan konfirmasi proses data perawat
 4. Kelola Data Obat
 - Admin memilih opsi "Kelola data obat"
 - Sistem memproses data obat dan informasi farmasi
 - Database mengkonfirmasi penyimpanan data obat berhasil
- Alur Kerja:
 1. Setelah login, admin mengakses menu kelola data
 2. Admin memilih jenis data yang akan dikelola (pasien, dokter, perawat, atau obat)

3. Sistem memproses data sesuai pilihan ke database
4. Database memberikan konfirmasi hasil proses
5. Admin dapat melanjutkan mengelola jenis data lain
6. Proses berakhir ketika admin selesai mengelola semua data yang diperlukan

b. Sequence diagram lihat data

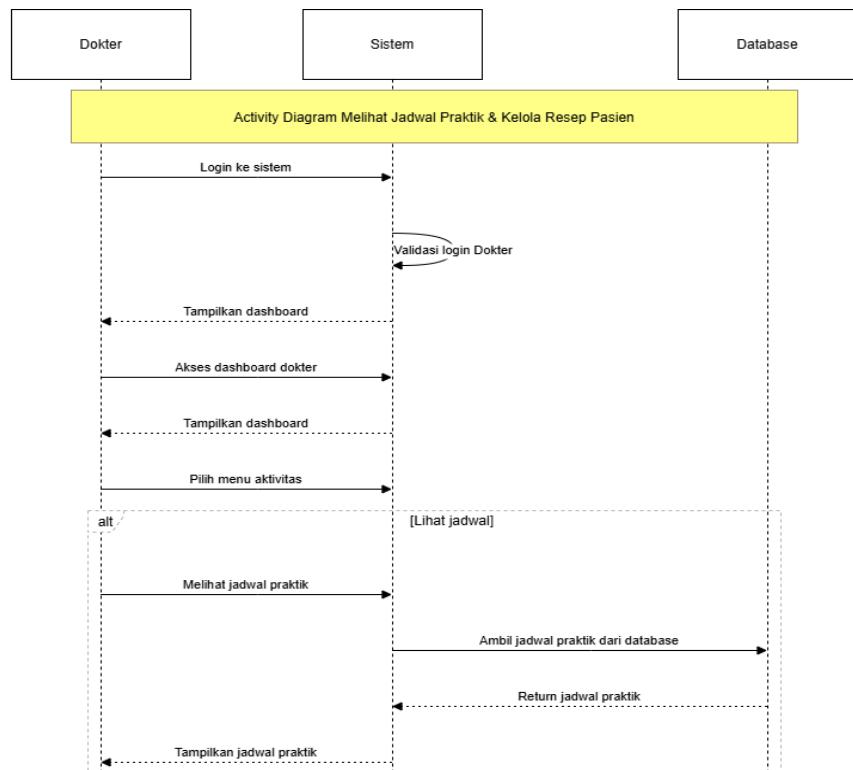




Sequence diagram ini menggambarkan alur interaksi antara tiga komponen utama sistem: Admin, Sistem, dan Database. Diagram ini menunjukkan proses lengkap mulai dari login admin hingga pengelolaan data dengan berbagai operasi CRUD (Create, Read, Update, Delete).

- Admin: Pengguna yang memiliki hak akses untuk mengelola sistem
- Sistem: Aplikasi atau interface yang memproses permintaan admin
- Database: Tempat penyimpanan data yang menyediakan layanan persistensi

c. Sequence diagram melihat jam praktik



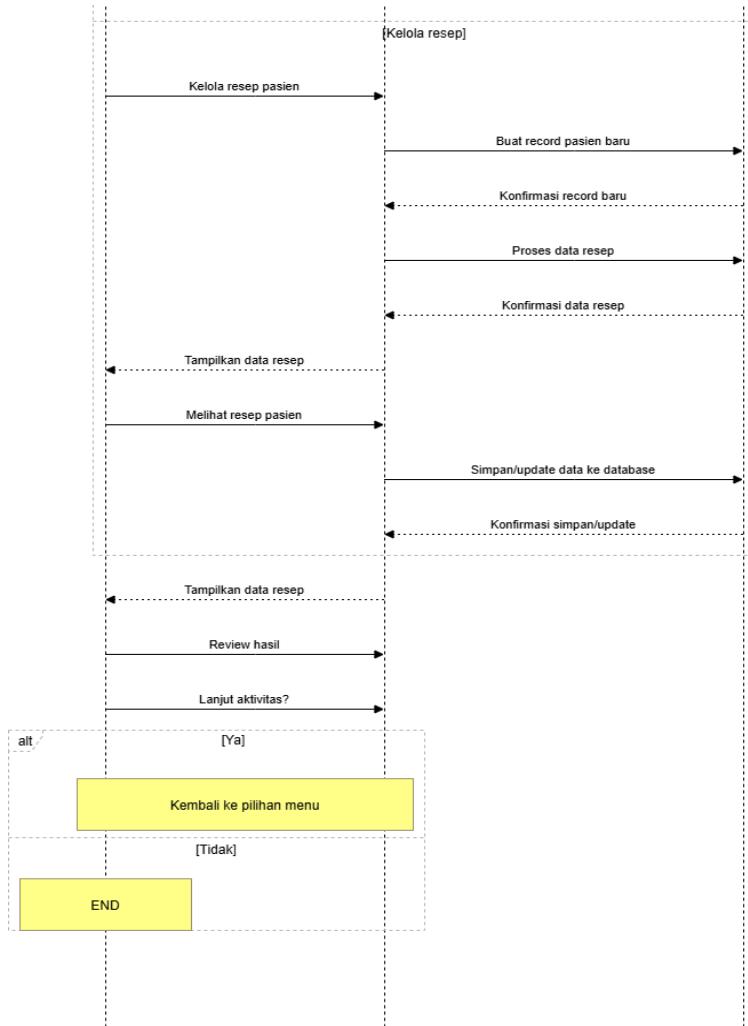


Diagram ini menjelaskan dua fungsi utama dokter dalam sistem:

a. Melihat Jadwal Praktik

1. Login Validasi: Dokter login ke sistem dengan validasi yang berhasil
2. Akses Dashboard: Sistem menampilkan dashboard dokter
3. Pilih Menu Aktivitas: Dokter memilih menu untuk melihat jadwal praktik
4. Ambil Data Jadwal: Sistem mengambil data jadwal praktik dari database
5. Tampilkan Jadwal: Sistem menampilkan jadwal praktik dokter

b. Kelola Resep Pasien

1. Pilih Kelola Resep: Dokter memilih menu kelola resep pasien
2. Buat Record Baru: Sistem memungkinkan pembuatan record resep baru
3. Konfirmasi Data: Sistem mengkonfirmasi data resep baru
4. Proses ke Database: Data resep diproses dan disimpan ke database
5. Update Konfirmasi: Database memberikan konfirmasi penyimpanan data
6. Tampilkan Hasil: Sistem menampilkan data resep yang telah disimpan
7. Review dan Kontinuitas: Dokter dapat melakukan review hasil dan melanjutkan aktivitas lain

d. Sequence diagram login admin

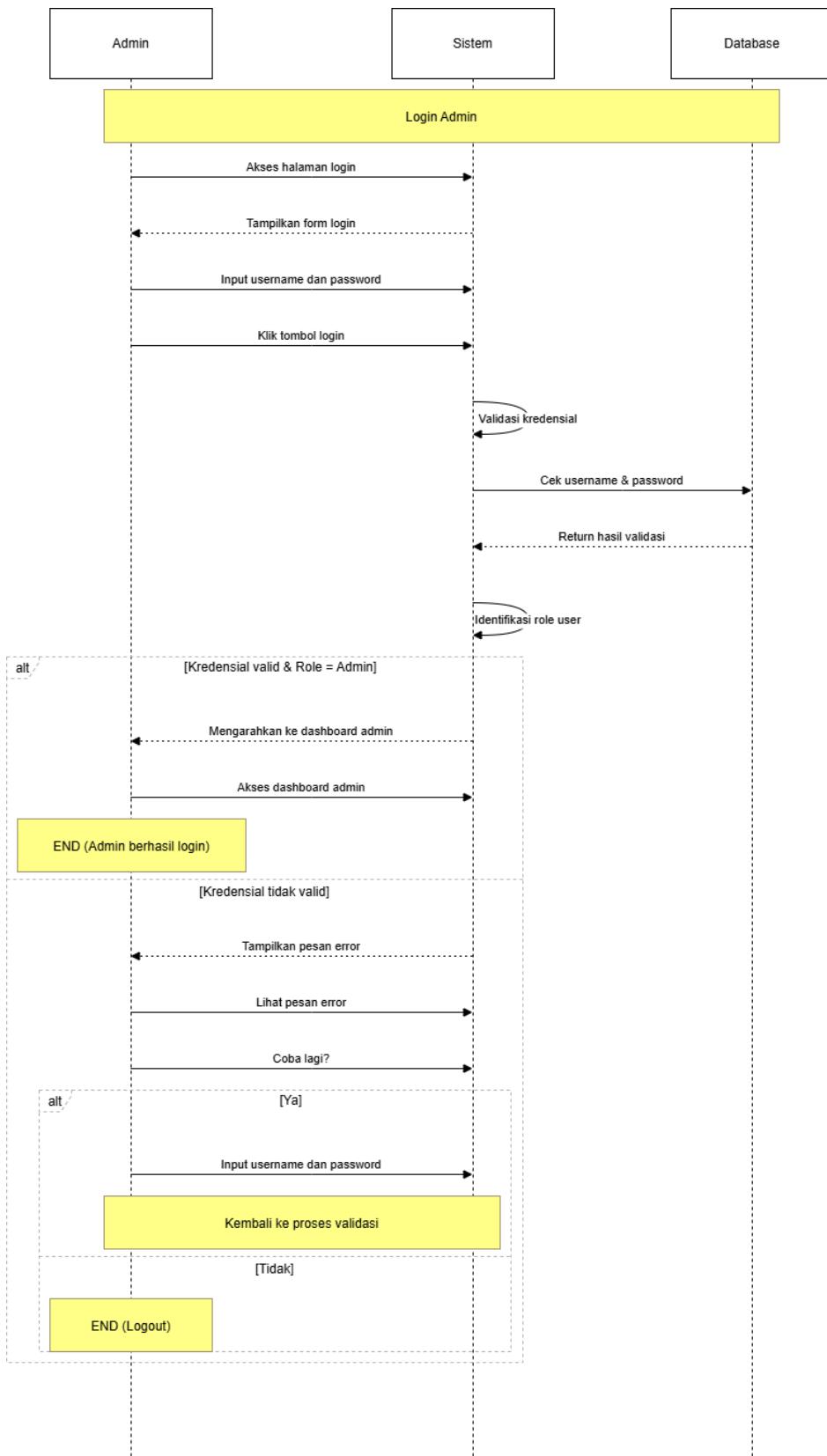
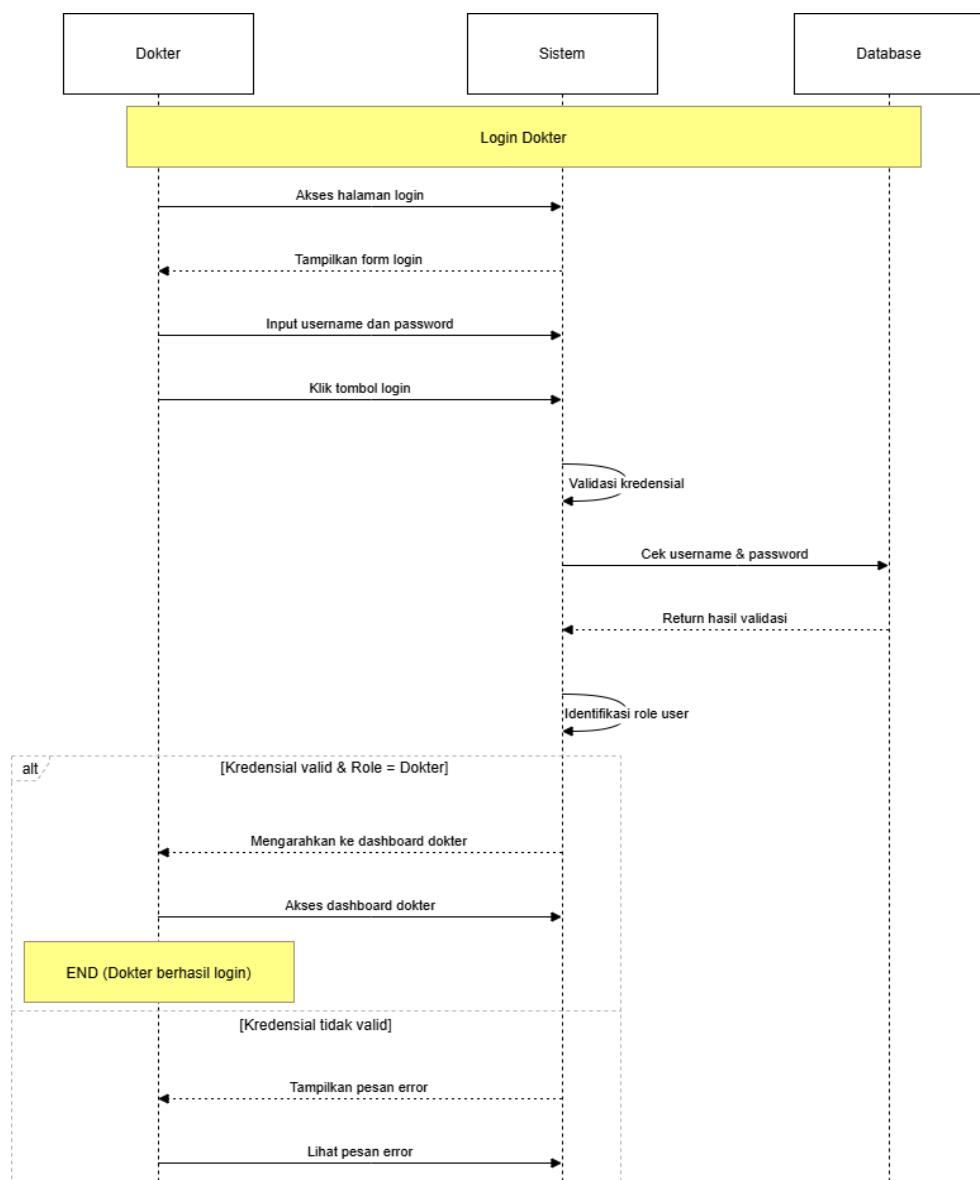


Diagram ini menunjukkan alur proses autentikasi untuk pengguna dengan role admin.

Prosesnya serupa dengan login dokter namun dengan akses ke dashboard admin:

1. Akses Halaman Login: Admin mengakses halaman login sistem
2. Input Kredensial: Admin memasukkan username dan password
3. Validasi Database: Sistem memverifikasi kredensial melalui database
4. Identifikasi Role: Sistem mengkonfirmasi role user sebagai admin
5. Redirect Dashboard: Jika berhasil, sistem mengarahkan ke dashboard admin
6. Error Handling: Penanganan kesalahan jika kredensial tidak valid
7. Logout Process: Proses logout dari sistem

e. Sequence diagram login dokter



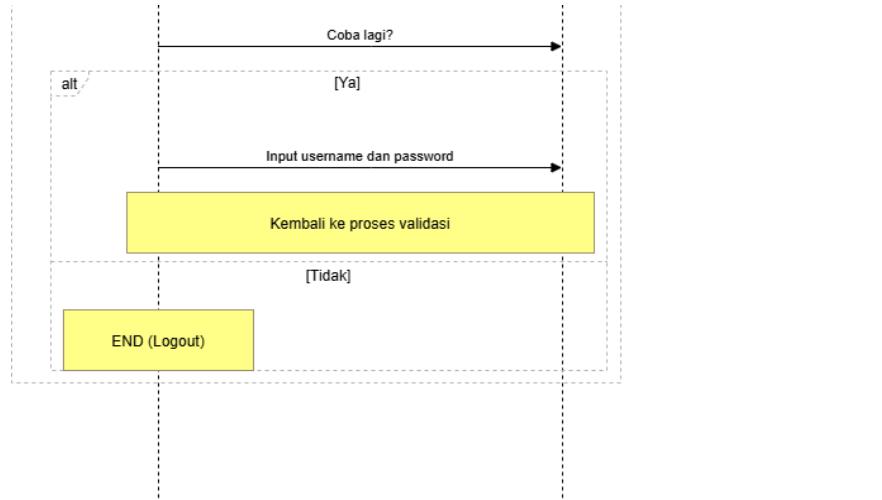


Diagram ini menggambarkan alur proses autentikasi untuk pengguna dengan role dokter. Proses dimulai ketika dokter mengakses halaman login sistem. Berikut adalah tahapan prosesnya:

1. Inisiasi Login: Dokter mengakses halaman login dan sistem menampilkan form login
2. Input Kredensial: Dokter memasukkan username dan password, kemudian mengklik tombol login
3. Validasi Kredensial: Sistem melakukan pengecekan username dan password ke database
4. Pengembalian Hasil: Database mengembalikan hasil validasi ke sistem
5. Identifikasi Role: Sistem mengidentifikasi role user sebagai dokter
6. Akses Dashboard: Jika kredensial valid dan role adalah dokter, sistem mengarahkan ke dashboard dokter
7. Penanganan Error: Jika kredensial tidak valid, sistem menampilkan pesan error dan meminta user untuk mencoba lagi
8. Logout: Proses berakhir dengan kemungkinan logout dari sistem

3.2 Implementasi

3.2.1 Folder Controller

Bagian ini akan fokus pada kelas-kelas dalam paket controller. Peran utama dari Controller dalam pola desain Model-View-Controller (MVC) adalah sebagai perantara antara View (antarmuka pengguna/GUI) dan Model (logika bisnis dan data/DAO). Controller menerima input dari View, memprosesnya (terkadang dengan validasi), dan kemudian memanggil Model untuk memperbarui data.

A. DokterController.java

```
1 package controller;
2
3 import dao.DokterDAO;
4 import model.Dokter;
5 import java.util.List;
6
7 /**
8 * Controller untuk mengelola data dokter, sebagai perantara antara view dan model.
9 * Menangani logika bisnis terkait operasi CRUD dokter.
10 */
11 public class DokterController {
12
13 /**
14 * Menambahkan dokter baru ke sistem.
15 * @param id ID dokter
16 * @param nama Nama lengkap dokter
17 * @param spesialisasi Bidang spesialisasi dokter
18 * @param idUser ID user wajib (wajib diisi)
19 * @throws IllegalArgumentException jika ID atau nama kosong
20 */
21 public void tambahDokter(String id, String nama, String spesialisasi, String idUser) {
22     if (id.isEmpty() || nama.isEmpty()) {
23         throw new IllegalArgumentException("ID dan nama wajib diisi");
24     }
25
26     Dokter dokter = new Dokter(id, nama, spesialisasi, idUser);
27     DokterDAO.insertDokter(dokter);
28 }
29
30 /**
31 * Memperbarui data dokter yang sudah ada.
32 * @param id ID dokter yang akan diupdate
33 * @param nama Nama baru dokter
34 * @param spesialisasi Spesialisasi baru
35 * @param idUser ID user terkait
36 */
37 public void updateDokter(String id, String nama, String spesialisasi, String idUser) {
38     Dokter dokter = new Dokter(id, nama, spesialisasi, idUser);
39     DokterDAO.updateDokter(dokter);
40 }
41
42 /**
43 * Menghapus dokter dari sistem berdasarkan ID.
44 * @param id ID dokter yang akan dihapus
45 */
46 public void hapusDokter(String id) {
47     DokterDAO.deleteDokter(id);
48 }
49
50 /**
51 * Mengambil daftar semua dokter yang berdaftar.
52 * @return List berisi semua objek Dokter
53 */
54 public List<Dokter> ambilSemuaDokter() {
55     return DokterDAO.getAllDokter();
56 }
57 }
```

Penjelasan DokterController.java

a. Deskripsi Umum

DokterController.java bertanggung jawab atas logika bisnis untuk operasi CRUD (Create, Read, Update, Delete) pada data dokter.

b. Fungsionalitas Utama

- Menambahkan dokter baru.
- Memperbarui data dokter.
- Menghapus dokter.
- Mengambil daftar semua dokter.

c. Komponen Penting (Code Breakdown)

Kelas ini memiliki struktur yang sangat mirip dengan PasienController, menunjukkan konsistensi dalam desain aplikasi.

- tambahDokter(...)*: Memvalidasi bahwa ID dan nama tidak kosong, membuat objek Dokter, dan memanggil DokterDAO.insertDokter(dokter).
- updateDokter(...)*: Membuat objek Dokter dengan data baru dan memanggil DokterDAO.updateDokter(dokter).
- hapusDokter(String id)*: Memanggil DokterDAO.deleteDokter(id) untuk menghapus dokter berdasarkan ID-nya.
- ambilSemuaDokter()*: Mengembalikan List<Dokter> dari DokterDAO.getAllDokter().

B. JadwalDokterController.java

```
5  package klinik_pbo;
6
7  import dao.JadwalDokterDAO;
8  import javax.swing.table.DefaultTableModel;
9  import klinik_pbo.koneksi;
10 import java.sql.Connection;
11 import java.sql.DriverManager;
12 import java.sql.SQLException;
13 import java.util.List;
14 import java.util.logging.Level;
15 import java.util.logging.Logger;
16 import javax.swing.JOptionPane;
17 import javax.swing.JOptionPanel;
18 import javax.swing.SwingUtilities;
19 import model.JadwalDokter;
20
21 /**
22 * @author avvjelly
23 */
24 public class JadwalDokterDokter extends javax.swing.JFrame {
25     public void tampilkanDataJadwalDokter() {
26         // Pastikan dijalankan di EDT
27         SwingUtilities.invokeLater(() -> {
28             try {
29                 // Ambil data dari DAO
30                 List<JadwalDokter> daftar = JadwalDokterDAO.getAllJadwalDokterLengkap();
31
32                 // Bersihkan tabel
33                 DefaultTableModel model = (DefaultTableModel) TabelJadwalDokter.getModel();
34                 model.setRowCount(0);
35
36                 // Jika data kosong, tampilkan pesan
37                 if (daftar.isEmpty()) {
38                     JOptionPane.showMessageDialog(this, "Tidak ada data jadwal dokter.", "Info", JOptionPane.INFORMATION_MESSAGE);
39                     return;
40                 }
41
42                 // Isi tabel
43                 for (JadwalDokter jd : daftar) {
44                     model.addRow(new Object[]{
45                         jd.getIdJadwalDokter(),
46                         jd.getNamaDokter(),
47                         jd.getHari(),
```

```
48 |         jd.getNamaRuangan()
49 |     );
50 |
51 |     // Paksa update UI
52 |     TabelJadwalDokter.revalidate();
53 |     TabelJadwalDokter.repaint();
54 |
55 | } catch (Exception e) {
56 |     JOptionPane.showMessageDialog(this, "Gagal memuat data: " + e.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
57 |     e.printStackTrace();
58 | }
59 |
60 |
61 |
62 |
63 |
64 | /**
65 | * Creates new form JadwalDokterDokter
66 | */
67 | public JadwalDokterDokter() {
68 |     initComponents();
69 |     tampilkanDataJadwalDokter();
70 |
71 |
72 | /**
73 | * This method is called from within the constructor to initialize the form.
74 | * WARNING: Do NOT modify this code. The content of this method is always
75 | * regenerated by the Form Editor.
76 | */
77 | @SuppressWarnings("unchecked")
78 | Generated Code
79 |
80 | private void ButtonDashboardActionPerformed(java.awt.event.ActionEvent evt) {
81 |     // TODO add your handling code here:
82 |     // 1. Tutup form saat ini
83 |     this.dispose();
84 |
85 |     // 2. Buka form
86 |     DashboardDokter dashboard = new DashboardDokter();
87 |     dashboard.setVisible(true);
88 |
89 |
90 |
91 |
92 |
93 |
94 |
95 |
96 |
97 |
98 |
99 |
100 |
101 |
102 |
103 |
104 |
105 |
106 |
107 |
108 |
109 |
110 |
111 |
112 |
113 |
114 |
115 |
116 |
117 |
118 |
119 |
120 |
121 |
122 |
123 |
124 |
125 |
126 |
127 |
128 |
129 |
130 |
131 |
132 |
133 |
134 |
135 |
136 |
137 |
138 |
139 |
140 |
141 |
142 |
143 |
144 |
145 |
146 |
147 |
148 |
149 |
150 |
151 |
152 |
153 |
154 |
155 |
156 |
157 |
158 |
159 |
160 |
161 |
162 |
163 |
164 |
165 |
166 |
167 |
168 |
169 |
170 |
171 |
172 |
173 |
174 |
175 |
176 |
177 |
178 |
179 |
180 |
181 |
182 |
183 |
184 |
185 |
186 |
187 |
188 |
189 |
190 |
191 |
192 |
193 |
194 |
195 |
196 |
197 |
198 |
199 |
200 |
201 |
202 |
203 |
204 |
205 |
206 |
207 |
208 |
209 |
210 |
211 |
212 |
213 |
214 |
215 |
216 |
217 |
218 |
219 |
220 |
221 |
222 |
223 |
224 |
225 |
226 |
227 |
228 |
229 |
230 |
231 |
232 |
233 |
234 |
235 |
236 |
237 |
238 |
239 |
240 |
241 |
242 |
243 |
244 |
245 |
246 |
247 |
248 |
249 |
250 |
251 |
252 |
253 |
254 |
255 |
256 |
257 |
258 |
259 |
260 |
261 |
262 |
263 |
264 |
265 |
266 |
267 |
268 |
269 |
270 |
271 }
```

```
234 | private void ButtonJadwalDokterActionPerformed(java.awt.event.ActionEvent evt) {
235 |     // TODO add your handling code here:
236 |     // 1. Tutup form saat ini
237 |     this.dispose();
238 |
239 |     // 2. Buka form
240 |     JadwalDokterDokter daftarjadwal = new JadwalDokterDokter();
241 |     daftarjadwal.setVisible(true);
242 |
243 |
244 | private void ButtonResepPasienActionPerformed(java.awt.event.ActionEvent evt) {
245 |     // TODO add your handling code here:
246 |     // 1. Tutup form saat ini
247 |     this.dispose();
248 |
249 |     // 2. Buka form
250 |     ResepPasienDokter daftardesep = new ResepPasienDokter();
251 |     daftardesep.setVisible(true);
252 |
253 |
254 | private void ButtonLogoutActionPerformed(java.awt.event.ActionEvent evt) {
255 |     // TODO add your handling code here:
256 |     // Tampilkan dialog konfirmasi
257 |     int pilihan = JOptionPane.showConfirmDialog(this,
258 |         "Apakah Anda yakin ingin logout?",
259 |         "Konfirmasi Logout",
260 |         JOptionPane.YES_NO_OPTION);
261 |
262 |     // Jika pengguna memilih "Yes"
263 |     if (pilihan == JOptionPane.YES_OPTION) {
264 |         // 1. Tutup form saat ini
265 |         this.dispose();
266 |
267 |         // 2. Buka form login
268 |         Login login = new Login();
269 |         login.setVisible(true);
270 |
271 |     }
```

Penjelasan JadwalDokterController.java

a. Deskripsi Umum

JadwalDokterController.java` adalah controller yang lebih interaktif dan kompleks. Kelas ini tidak hanya meneruskan data, tetapi juga membangun antarmuka dialog dinamis menggunakan `JOptionPane` untuk mengelola jadwal dokter.

b. Fungsionalitas Utama

- Menampilkan Dialog Aksi: Menyajikan pilihan kepada admin untuk "Insert", "Edit", atau "Delete" jadwal.
- Create (Insert): Membangun formulir input dinamis menggunakan `JOptionPane` yang berisi `JTextField` dan `JComboBox` untuk memilih dokter dan ruangan. Data untuk `JComboBox` diambil dari DAO (`getAllDokterForCombo`, `getAllRuanganForCombo`).
- Update (Edit): Memungkinkan admin memilih jadwal yang ada dari `JComboBox`, lalu menampilkan formulir edit yang sudah terisi data lama.
- Delete: Memungkinkan admin memilih jadwal dari `JComboBox` dan, setelah konfirmasi, menghapusnya.

c. Komponen Penting (Code Breakdown)

- `showAlertDialog()`: Metode publik utama yang dipanggil oleh View ('JadwalDokterAdmin'). Metode ini menampilkan ` JOptionPane.showOptionDialog` dan menggunakan `switch` untuk mengarahkan alur program ke metode yang sesuai ('tambahJadwalDokter', 'editJadwalDokter', 'hapusJadwalDokter').
- Metode `private` ('tambahJadwalDokter', 'editJadwalDokter', 'hapusJadwalDokter'): Masing-masing metode ini merangkum logika untuk satu operasi CRUD. Mereka bertanggung jawab untuk membangun dialog, mengambil input pengguna, memvalidasi data, dan memanggil metode DAO yang relevan.
- `handleError(...)`: Sebuah private helper method untuk menangani `Exception` (terutama `SQLException`) dengan cara yang konsisten: mencetak stack trace dan menampilkan pesan error kepada pengguna.

C. ObatController.java

```
1 package controller;
2
3 import dao.ObatDAO;
4 import model.Obat;
5 import java.util.List;
6
7 /**
8 * Controller untuk manajemen data obat.
9 * Menangani logika bisnis terkait operasi CRUD obat.
10 */
11 public class ObatController {
12
13     /**
14      * Menambahkan obat baru.
15      * @param id ID obat
16      * @param nama Nama obat
17      * @param stok Stok awal
18      * @param harga Harga per unit
19      * @throws IllegalArgumentException jika ID atau nama kosong
20      */
21     public void tambahObat(String id, String nama, int stok, double harga) {
22         if (id.isEmpty() || nama.isEmpty()) {
23             throw new IllegalArgumentException("ID dan nama obat wajib diisi");
24         }
25
26         Obat obat = new Obat(id, nama, stok, harga);
27         ObatDAO.insertObat(obat);
28     }
29
30     /**
31      * Mengupdate data obat yang sudah ada.
32      * @param id ID obat yang akan diupdate
33      * @param nama Nama baru
34      * @param stok Stok baru
35      * @param harga Harga baru
36      */
37     public void updateObat(String id, String nama, int stok, double harga) {
38         Obat obat = new Obat(id, nama, stok, harga);
39         ObatDAO.updateObat(obat);
40     }
41
42     /**
43      * Menghapus obat berdasarkan ID.
44      * @param id ID obat yang akan dihapus
45      */
46     public void hapusObat(String id) {
47         ObatDAO.deleteObat(id);
48     }
49
50     /**
51      * Mengambil daftar semua obat.
52      * @return List berisi semua objek Obat
53      */
54     public List<Obat> ambilSemuaObat() {
55         return ObatDAO.getAllObat();
56     }
57 }
```

Penjelasan ObatController.java

a. Deskripsi Umum

'ObatController.java' mengelola logika bisnis untuk data obat-obatan di klinik.

b. Fungsionalitas Utama

Menangani penambahan, pembaruan, penghapusan, dan pengambilan data obat.

c. Komponen Penting (Code Breakdown)

- `tambahObat(...)`: Menerima detail obat, melakukan validasi pada ID dan nama, membuat objek 'Obat', dan memanggil 'ObatDAO.insertObat(obat)' .

- `updateObat(...)`: Membuat objek 'Obat' dengan data yang telah diubah dan memanggil `ObatDAO.updateObat(obat)`.
- `hapusObat(String id)`: Memanggil `ObatDAO.deleteObat(id)`.
- `ambilSemuaObat()`: Mengembalikan `List<Obat>` dari `ObatDAO.getAllObat()`.

D. PasienController.java

```

1  package controller;
2
3  import dao.PasienDAO;
4  import model.Pasien;
5  import java.util.List;
6
7  /**
8   * Controller untuk manajemen data pasien.
9   * Menangani logika bisnis terkait operasi CRUD pasien.
10  */
11 public class PasienController {
12
13     /**
14      * Menambahkan pasien baru ke sistem.
15      * @param id ID pasien (unik)
16      * @param nama Nama lengkap pasien
17      * @param alamat Alamat rumah
18      * @param telepon Nomor telepon pasien
19      * @throws IllegalArgumentException jika ID atau nama kosong
20     */
21     public void tambahPasien(String id, String nama, String alamat, String telepon) {
22         if (id.isEmpty() || nama.isEmpty()) {
23             throw new IllegalArgumentException("ID dan nama wajib diisi");
24         }
25
26         Pasien pasien = new Pasien(id, nama, alamat, telepon);
27         PasienDAO.insertPasien(pasien);
28     }
29
30     /**
31      * Memperbarui data pasien yang sudah ada.
32      * @param id ID pasien yang akan diupdate
33      * @param nama Nama baru pasien
34      * @param alamat Alamat baru
35      * @param telepon Nomor telepon baru
36     */
37     public void updatePasien(String id, String nama, String alamat, String telepon) {
38         Pasien pasien = new Pasien(id, nama, alamat, telepon);
39         PasienDAO.updatePasien(pasien);
40     }
41
42     /**
43      * Menghapus pasien berdasarkan ID.
44      * @param id ID pasien yang akan dihapus
45     */
46     public void hapusPasien(String id) {
47         PasienDAO.deletePasien(id);
48     }
49
50     /**
51      * Mengambil daftar semua pasien yang terdaftar.
52      * @return List berisi semua objek Pasien
53     */
54     public List<Pasien> ambilSemuaPasien() {
55         return PasienDAO.getAllPasien();
56     }
57 }
```

Penjelasan PasienController.java

a. Deskripsi Umum

'PasienController.java' adalah kelas yang menangani semua logika bisnis yang terkait dengan manajemen data pasien. Kelas ini menjadi perantara antara antarmuka pendaftaran/daftar pasien dan 'PasienDAO'.

b. Fungsionalitas Utama

- Create: Menambahkan pasien baru ke database.
- Update: Memperbarui informasi pasien yang sudah ada.
- Delete: Menghapus data pasien dari database.
- Read: Mengambil daftar semua pasien.

c. Komponen Penting (Code Breakdown)

- 'tambahPasien(...)': Menerima data mentah (ID, nama, alamat, telepon) dari View. Metode ini melakukan validasi untuk memastikan ID dan nama tidak kosong, dan jika validasi gagal, ia akan melempar 'IllegalArgumentException'. Jika data valid, ia membuat objek 'Pasien' baru dan meneruskannya ke 'PasienDAO.insertPasien(pasien)' untuk disimpan ke database.
- 'updatePasien(...)': Menerima data pasien yang diperbarui, membuat objek 'Pasien', dan memanggil 'PasienDAO.updatePasien(pasien)'.
- 'hapusPasien(String id)': Menerima ID pasien dan langsung memanggil 'PasienDAO.deletePasien(id)' untuk melakukan penghapusan.
- 'ambilSemuaPasien()': Memanggil 'PasienDAO.getAllPasien()' dan mengembalikan sebuah 'List<Pasien>' yang akan digunakan oleh View untuk ditampilkan dalam tabel.

E. ResepPasienController.java

```
1 package controller;
2
3 import dao.ResepPasienDAO;
4 import model.ResepPasien;
5 import javax.swing.*;
6 import java.sql.Date;
7 import java.util.*;
8
9 /**
10 * Controller untuk manajemen resep pasien.
11 * Kegiatan pada controller berbasis role (admin/dokter).
12 */
13 public class ResepPasienController {
14     private final ResepPasienDAO dao;
15     private final String role; // admin / dokter
16
17     /**
18      * Konstruktor controller.
19      * param role Role pengguna (admin/dokter) untuk menentukan aksi yang cocok
20      */
21     public ResepPasienController(String role) {
22         this.dao = new ResepPasienDAO();
23         this.role = role;
24     }
25
26     /**
27      * Menampilkan dialog aksi berdasarkan role pengguna.
28      */
29     public void showActionDialog() {
30         if (role.equals("admin")) {
31             showAdminActions();
32         } else {
33             showDokterActions();
34         }
35     }
36
37     private void showAdminActions() {
38         Object[] options = {"Ubah Status", "Hapus", "Batal"};
39         int pilihan = JOptionPane.showOptionDialog(null,
40             "Pilih Aksi Admin",
41             "Manajemen Resep (Admin)",
```

```

42 |         JOptionPane.DEFAULT_OPTION,
43 |         JOptionPane.INFORMATION_MESSAGE,
44 |         null,
45 |         options,
46 |         options[0]);
47 |
48 |     switch (pilihan) {
49 |     case 0 -> showAdminStatusUpdateDialog();
50 |     case 1 -> hapusResep();
51 |     }
52 |
53 | }
54 | 
55 | private void showDokterActions () {
56 |     Object[] options = {"Insert", "Edit", "Batal"};
57 |     int pilihan = JOptionPane.showOptionDialog(null,
58 |         "Pilih Aksi untuk Resep Pasien",
59 |         "Manajemen Resep (Dokter)",
60 |         JOptionPane.DEFAULT_OPTION,
61 |         JOptionPane.INFORMATION_MESSAGE,
62 |         null,
63 |         options,
64 |         options[0]);
65 |
66 |     switch (pilihan) {
67 |     case 0 -> tambahResep();
68 |     case 1 -> editResep();
69 |     }
70 |
71 | /**
72 | * Menampilkan dialog untuk mengubah status resep (admin only).
73 | */
74 | private void showAdminStatusUpdateDialog () {
75 |     try {
76 |         List<ResepPasien> resepList = dao.getAllResepLengkap();
77 |         if (resepList.isEmpty()) {
78 |             JOptionPane.showMessageDialog(null, "Tidak ada resep untuk diubah status.");
79 |             return;
80 |         }
81 |
82 |         Map<String, String> resepMap = createResepSelectionMap(resepList);
83 |         String idResep = showSelectionDialog(resepMap, "Pilih Resep");
84 |         if (idResep == null) return;

```

```

86 | 
87 |         Map<String, String> data = dao.getResepById(idResep);
88 |         String newStatus = showStatusUpdateDialog(data.get("status"));
89 |         if (newStatus != null) {
90 |             dao.updateStatusResep(idResep, newStatus);
91 |             JOptionPane.showMessageDialog(null, "Status berhasil diperbarui.");
92 |             refreshTable();
93 |         }
94 |     } catch (Exception e) {
95 |         handleError("Update Status Error", e);
96 |     }
97 |
98 | /**
99 | * Menambahkan resep baru (dokter only).
100 | */
101 | private void tambahResep () {
102 |     try {
103 |         // Prepare form components
104 |         Map<String, String> pasienMap = dao.getAllPasienForCombo();
105 |         Map<String, String> dokterMap = dao.getAllDokterForCombo();
106 |         Map<String, String> obatMap = dao.getAllObatForCombo();
107 |
108 |         JTextField idResepField = new JTextField();
109 |         JComboBox<String> pasienCombo = new JComboBox<String>(pasienMap.keySet().toArray(new String[0]));
110 |         JComboBox<String> dokterCombo = new JComboBox<String>(dokterMap.keySet().toArray(new String[0]));
111 |         JComboBox<String> obatCombo = new JComboBox<String>(obatMap.keySet().toArray(new String[0]));
112 |         JTextField penyakitField = new JTextField();
113 |         JSpinner tanggalSpinner = createDateSpinner();
114 |
115 |         // Show input dialog
116 |         if (showResepInputDialog(idResepField, pasienCombo, dokterCombo, obatCombo, penyakitField, tanggalSpinner)) {
117 |             String idResep = validateResepId(idResepField.getText().trim());
118 |             if (idResep == null) return;
119 |
120 |             // Collect input data
121 |             ResepData resepData = collectResepData(pasienMap, dokterMap, obatMap,
122 |                 pasienCombo, dokterCombo, obatCombo, penyakitField, tanggalSpinner);
123 |
124 |             // Insert new resep
125 |             dao.insertResep(idResep, resepData.penyakit, resepData.tanggal,
126 |                 resepData.idPasien, resepData.idDokter, resepData.idObat, "Belum ditebus");

```

```

128     | JOptionPane.showMessageDialog(null, "Resep berhasil ditambahkan.");
129     | refreshTable();
130 } catch (Exception e) {
131     | handleError("Insert Error", e);
132 }
133 }
134
135 /**
136 * Mengedit resep yang sudah ada (dokter only).
137 */
138 private void editResep() {
139     try {
140         List<ResepPasien> resepList = dao.getAllResepLengkap();
141         if (resepList.isEmpty()) {
142             JOptionPane.showMessageDialog(null, "Tidak ada resep yang tersedia untuk diedit.");
143             return;
144         }
145
146         Map<String, String> resepMap = createResepSelectionMap(resepList);
147         String idResep = showSelectionDialog(resepMap, "Pilih Resep untuk Diedit");
148         if (idResep == null) return;
149
150         Map<String, String> data = dao.getResepById(idResep);
151         if (showResepEditDialog(data, idResep)) {
152             refreshTable();
153         }
154     } catch (Exception e) {
155         handleError("Edit Error", e);
156     }
157 }
158
159 /**
160 * Menghapus resep (admin only).
161 */
162 private void hapusResep() {
163     try {
164         List<ResepPasien> list = dao.getAllResepLengkap();
165         if (list.isEmpty()) {
166             JOptionPane.showMessageDialog(null, "Tidak ada resep yang tersedia.");
167             return;
168         }
169     }
170
171     Map<String, String> resepMap = createResepSelectionMap(list);
172     String idResep = showSelectionDialog(resepMap, "Pilih Resep untuk Dihapus");
173     if (idResep == null) return;
174
175     if (confirmDelete()) {
176         dao.deleteResep(idResep);
177         JOptionPane.showMessageDialog(null, "Resep berhasil dihapus.");
178         refreshTable();
179     }
180 } catch (Exception e) {
181     handleError("Delete Error", e);
182 }
183 }
184
185 // Helper methods
186 private Map<String, String> createResepSelectionMap(List<ResepPasien> resepList) {
187     Map<String, String> map = new LinkedHashMap<>();
188     for (ResepPasien r : resepList) {
189         String label = r.getNamaPasien() + " - " + r.getNamaDokter() + " (" + r.getIdResep() + ")";
190         map.put(label, r.getIdResep());
191     }
192     return map;
193 }
194
195 private String showSelectionDialog(Map<String, String> map, String title) {
196     JComboBox<String> combo = new JComboBox<>(map.keySet().toArray(new String[0]));
197     int result = JOptionPane.showConfirmDialog(null, combo, title, JOptionPane.OK_CANCEL_OPTION);
198     return (result == JOptionPane.OK_OPTION) ? map.get(combo.getSelectedItem()) : null;
199 }
200
201 private String showStatusUpdateDialog(String currentStatus) {
202     JComboBox<String> statusCombo = new JComboBox<>(new String[]{"belum ditebus", "ditebus"});
203     statusCombo.setSelectedItem(currentStatus);
204
205     Object[] field = {"Status:", statusCombo};
206     int konfirmasi = JOptionPane.showConfirmDialog(null, field, "Ubah Status Resep", JOptionPane.OK_CANCEL_OPTION);
207     return (konfirmasi == JOptionPane.OK_OPTION) ? (String) statusCombo.getSelectedItem() : null;
208 }
209
210 private JSpinner createDateSpinner() {
211     JSpinner spinner = new JSpinner(new SpinnerDateModel());
212     spinner.setEditor(new JSpinner.DateEditor(spinner, "yyyy-MM-dd"));
213     return spinner;

```

```

214 }
215
216 private boolean showResepInputDialog(JTextField idField, JComboBox<String> pasienCombo,
217     JComboBox<String> dokterCombo, JComboBox<String> obatCombo,
218     JTextField penyakitField, JSpinner tanggalSpinner) {
219
220     Object[] fields = {
221         "ID Resep:", idField,
222         "Pasien:", pasienCombo,
223         "Dokter:", dokterCombo,
224         "Obat:", obatCombo,
225         "Penyakit:", penyakitField,
226         "Tanggal:", tanggalSpinner
227     };
228
229     return JOptionPane.showConfirmDialog(null, fields, "Tambah Resep", JOptionPane.OK_CANCEL_OPTION) == JOptionPane.OK_OPTION;
230 }
231
232 private String validateResepId(String id) {
233     if (id.isEmpty()) {
234         JOptionPane.showMessageDialog(null, "ID Resep wajib diisi!", "Error", JOptionPane.ERROR_MESSAGE);
235         return null;
236     }
237     return id;
238 }
239
240 private ResepData collectResepData(Map<String, String> pasienMap, Map<String, String> dokterMap,
241     Map<String, String> obatMap, JComboBox<String> pasienCombo,
242     JComboBox<String> dokterCombo, JComboBox<String> obatCombo,
243     JTextField penyakitField, JSpinner tanggalSpinner) {
244
245     ResepData data = new ResepData();
246     data.idPasien = pasienMap.get(pasienCombo.getSelectedItem());
247     data.idDokter = dokterMap.get(dokterCombo.getSelectedItem());
248     data.idObat = obatMap.get(obatCombo.getSelectedItem());
249     data.penyakit = penyakitField.getText();
250     data.tanggal = new Date((java.util.Date) tanggalSpinner.getValue().getTime());
251     return data;
252 }
253
254 private boolean showResepEditDialog(Map<String, String> data, String idResep) throws Exception {
255     // Prepare form components with existing data
256     Map<String, String> pasienMap = dao.getAllPasienForCombo();

```

```

256
257     Map<String, String> pasienMap = dao.getAllPasienForCombo();
258     Map<String, String> dokterMap = dao.getAllDokterForCombo();
259     Map<String, String> obatMap = dao.getAllObatForCombo();
260
261     JComboBox<String> pasienCombo = new JComboBox<>(pasienMap.keySet().toArray(new String[0]));
262     pasienCombo.setSelectedItem(data.get("namaPasien"));
263
264     JComboBox<String> dokterCombo = new JComboBox<String>(dokterMap.keySet().toArray(new String[0]));
265     dokterCombo.setSelectedItem(data.get("namaDokter"));
266
267     JComboBox<String> obatCombo = new JComboBox<String>(obatMap.keySet().toArray(new String[0]));
268     obatCombo.setSelectedItem(data.get("namaObat"));
269
270     JTextField penyakitField = new JTextField(data.get("penyakit"));
271     JSpinner tanggalSpinner = createDateSpinner();
272     tanggalSpinner.setValue(Date.valueOf(data.get("tanggal")));
273
274     // Show edit dialog
275     Object[] fields = {
276         "Pasien:", pasienCombo,
277         "Dokter:", dokterCombo,
278         "Obat:", obatCombo,
279         "Penyakit:", penyakitField,
280         "Tanggal:", tanggalSpinner
281     };
282
283     if (JOptionPane.showConfirmDialog(null, fields, "Edit Resep", JOptionPane.OK_CANCEL_OPTION) == JOptionPane.OK_OPTION) {
284         ResepData resepData = collectResepData(pasienMap, dokterMap, obatMap,
285             pasienCombo, dokterCombo, obatCombo, penyakitField, tanggalSpinner);
286
287         dao.updateResepTanpaStatus(idResep, resepData.penyakit, resepData.tanggal,
288             resepData.idPasien, resepData.idDokter, resepData.idObat);
289
290         JOptionPane.showMessageDialog(null, "Resep berhasil diperbarui.");
291         return true;
292     }
293     return false;
294 }

```

```

295     private boolean confirmDelete() {
296         return JOptionPane.showConfirmDialog(null,
297             "Yakin ingin menghapus resep ini?", "Konfirmasi",
298             JOptionPane.YES_NO_OPTION) == JOptionPane.YES_OPTION;
299     }
300
301     private void refreshTable() {
302         // Implementasi refresh tabel view
303     }
304
305     private void handleError(String title, Exception e) {
306         e.printStackTrace();
307         JOptionPane.showMessageDialog(null, "Terjadi kesalahan: " + e.getMessage(), title, JOptionPane.ERROR_MESSAGE);
308     }
309
310     // Helper class untuk mengelompokkan data resep
311     private static class ResepData {
312         String idPasien;
313         String idDokter;
314         String idObat;
315         String penyakit;
316         Date tanggal;
317     }
318 }

```

Penjelasan ResepPasienController.java

a. Deskripsi Umum

'ResepPasienController.java' adalah controller paling canggih dalam aplikasi ini, karena mengimplementasikan logika berbasis peran (role-based) untuk manajemen resep pasien.

b. Fungsionalitas Utama

- Kontrol Akses Berbasis Peran: Controller ini menerima parameter 'role' ("admin" atau "dokter") saat diinisialisasi. Aksi yang tersedia akan berbeda tergantung pada peran pengguna.
- Aksi Admin: Admin dapat mengubah status resep (misalnya dari "belum ditebus" menjadi "ditebus") dan menghapus resep.
- Aksi Dokter: Dokter dapat membuat (insert) resep baru dan mengedit resep yang sudah ada.
- Manajemen Dialog Dinamis: Sama seperti 'JadwalDokterController', kelas ini secara ekstensif menggunakan 'JOptionPane' untuk membuat dialog input dan pilihan yang kompleks.

c. Komponen Penting (Code Breakdown)

- `showActionDialog()`: Metode ini memeriksa nilai dari variabel 'role'. Jika "admin", ia akan memanggil `showAdminActions()`. Jika tidak, ia memanggil `showDokterActions()`.
- `showAdminActions()` vs `showDokterActions()`: Dua metode privat ini menampilkan dialog opsi yang berbeda sesuai dengan peran pengguna, mengimplementasikan logika kontrol akses.
- Metode CRUD ('tambahResep', 'editResep', 'hapusResep', 'showAdminStatusUpdateDialog'): Setiap metode ini mengelola satu alur kerja

spesifik, mulai dari menampilkan dialog, mengambil data, hingga memanggil DAO.

- Helper Methods: Kelas ini terstruktur dengan baik menggunakan banyak helper method privat, seperti:
 - `createResepSelectionMap(...)`: Untuk membuat `Map` yang akan digunakan untuk mengisi `JComboBox` pilihan resep.
 - `showSelectionDialog(...)`: Mengurangi duplikasi kode untuk menampilkan dialog pilihan `JComboBox`.
 - `collectResepData(...)`: Mengumpulkan data dari berbagai komponen form menjadi satu objek.
- `ResepData` Inner Class: Sebuah kelas `private static` sederhana yang digunakan sebagai struktur data untuk membawa informasi resep di dalam controller, membuat kode lebih bersih dan terorganisir.

F. RuanganController.java

```
5  package controller;
6
7  import dao.RuanganDAO;
8  import model.Ruangan;
9  import java.util.List;
10
11 public class RuanganController {
12
13     public void tambahRuangan(String id, String nama, int lantai, int kapasitas) {
14         if (id.isEmpty() || nama.isEmpty()) {
15             System.out.println("ID dan nama ruangan wajib diisi.");
16             return;
17         }
18
19         Ruangan r = new Ruangan(id, nama, lantai, kapasitas);
20         RuanganDAO.insertRuangan(r);
21     }
22
23     public void updateRuangan(String id, String nama, int lantai, int kapasitas) {
24         Ruangan r = new Ruangan(id, nama, lantai, kapasitas);
25         RuanganDAO.updateRuangan(r);
26     }
27
28     /**
29      * Menghapus ruangan berdasarkan ID.
30      * @param id ID ruangan yang akan dihapus.
31      * @throws IllegalArgumentException jika ID kosong.
32      */
33     public void hapusRuangan(String id) {
34         RuanganDAO.deleteRuangan(id);
35     }
36
37     /**
38      * Mengambil daftar semua ruangan yang terdaftar.
39      * @return List berisi semua objek Ruangan.
40      */
41     public List<Ruangan> ambilSemuaRuangan() {
42         return RuanganDAO.getAllRuangan();
43     }
44 }
```

Penjelasan RuanganController.java

a. Deskripsi Umum

‘RuanganController.java’ berfungsi sebagai pengendali logika bisnis untuk entitas ruangan.

b. Fungsionalitas Utama

Menangani operasi CRUD untuk data ruangan.

c. Komponen Penting (Code Breakdown)

- ‘tambahRuang(...)' : Melakukan validasi sederhana untuk ID dan nama, kemudian membuat objek ‘Ruang’ dan memanggil ‘RuangDAO.insertRuang(r)’.
- ‘updateRuang(...)' : Membuat objek ‘Ruang’ dan memanggil ‘RuangDAO.updateRuang(r)’.
- ‘hapusRuang(String id)' : Memanggil ‘RuangDAO.deleteRuang(id)’.
- ‘ambilSemuaRuang()' : Mengembalikan ‘List<Ruang>’ dari ‘RuangDAO.getAllRuang()’.

G. UserController.java

```
1  /*
2  * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this
3  * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
4  */
5 package controller;
6
7 import java.util.List;
8 import model.User;
9 import dao.UserDAO;
10
11 public class UserController {
12     public static List<User> ambilSemuaUser() {
13         return UserDAO.getAllUsers();
14     }
15 }
16
17 
```

a. Deskripsi Umum

UserController.java adalah controller yang sangat sederhana yang tugasnya hanya sebagai jembatan untuk mengambil data pengguna.

b. Fungsionalitas Utama

Menyediakan sebuah metode statis untuk mengambil semua data pengguna dari sistem.

c. Komponen Penting (Code Breakdown)

* ambilSemuaUser(): Ini adalah satu-satunya metode dalam kelas ini. Metode ini bersifat statis dan secara langsung memanggil UserDAO.getAllUsers(). Ini berfungsi sebagai wrapper atau pembungkus, yang memungkinkan bagian lain dari aplikasi (misalnya, View) untuk berinteraksi dengan data pengguna tanpa harus mengetahui implementasi dari UserDAO.

3.2.2 Folder Data Access Object (DAO)

Bagian ini menguraikan kelas-kelas dalam paket `dao`. Data Access Object (DAO) adalah sebuah pola desain yang digunakan untuk memisahkan logika akses data dari logika bisnis aplikasi. Setiap kelas DAO di sini secara eksklusif bertanggung jawab untuk semua operasi database (Create, Read, Update, Delete - CRUD) yang terkait dengan satu entitas atau tabel tertentu.

A. DokterDAO.java



```
public static void updateDokter(Dokter d) {
    Connection conn = koneksi.getKoneksi();
    String sql = "UPDATE dokter SET nama=?, spesialisasi=?, idUser=? WHERE idDokter=?";
    try {
        PreparedStatement stmt = conn.prepareStatement(sql);
        stmt.setString(1, d.getNama());
        stmt.setString(2, d.getSpesialisasi());
        stmt.setString(3, d.getIdUser());
        stmt.setString(4, d.getIdDokter());
        stmt.executeUpdate();
        stmt.close();
    } catch (SQLException e) {
        e.printStackTrace();
    }
}

public static void deleteDokter(String idDokter) {
    Connection conn = koneksi.getKoneksi();
    String sql = "DELETE FROM dokter WHERE idDokter=?";

    try {
        PreparedStatement stmt = conn.prepareStatement(sql);
        stmt.setString(1, idDokter);
        stmt.executeUpdate();
        stmt.close();
    } catch (SQLException e) {
        e.printStackTrace();
    }
}

package dao;

import model.Dokter;
import klinik_pbo.koneksi;
import java.sql.*;
import java.util.*;

public class DokterDAO {
    public static List<Dokter> getAllDokter() {
        List<Dokter> list = new ArrayList<>();
        Connection conn = koneksi.getKoneksi();

        try {
            String sql = "SELECT * FROM dokter";
            PreparedStatement stmt = conn.prepareStatement(sql);
            ResultSet rs = stmt.executeQuery();

            while (rs.next()) {
                Dokter d = new Dokter(
                    rs.getString("idDokter"),
                    rs.getString("nama"),
                    rs.getString("spesialisasi"),
                    rs.getString("idUser")
                );
                list.add(d);
            }

            rs.close();
            stmt.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return list;
    }

    public static void insertDokter(Dokter d) {
        Connection conn = koneksi.getKoneksi();

        try {
            // Generate ID User otomatis jika kosong
            if(d.getIdUser() == null || d.getIdUser().isEmpty()) {
                String newUserId = generateNewUserId(); // Perbaikan typo 'jssring' menjadi 'String'
                d.setIdUser(newUserId);
            }
            PreparedStatement stmt = conn.prepareStatement("INSERT INTO dokter (nama, spesialisasi, idUser) VALUES (?, ?, ?)");
            stmt.setString(1, d.getNama());
            stmt.setString(2, d.getSpesialisasi());
            stmt.setString(3, d.getIdUser());
            stmt.executeUpdate();
            stmt.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

```

41     public static void insertDokter(Dokter d) {
42         Connection conn = koneksi.getKoneksi();
43
44         try {
45             // Generate ID User otomatis jika kosong
46             if(d.getIdUser() == null || d.getIdUser().isEmpty()) {
47                 String newUserId = generateNewUserId(); // Perbaikan typo 'jserring' menjadi 'String'
48                 d.setIdUser(newUserId);
49             }
50
51             String sql = "INSERT INTO dokter (idDokter, nama, spesialisasi, idUser) VALUES (?, ?, ?, ?)" // Tambahkan tutup kurung
52
53             PreparedStatement stmt = conn.prepareStatement(sql);
54             stmt.setString(1, d.getIdDokter());
55             stmt.setString(2, d.getNama());
56             stmt.setString(3, d.getSpesialisasi());
57             stmt.setString(4, d.getIdUser());
58             stmt.executeUpdate();
59             stmt.close();
60
61         } catch (SQLException e) {
62             e.printStackTrace();
63         }
64     }
65
66     // Tambahkan method generateNewUserId()
67     private static String generateNewUserId() throws SQLException {
68         Connection conn = koneksi.getKoneksi();
69         String sql = "SELECT CONCAT('USR', LPAD(IFNULL(MAX(CAST(SUBSTRING(idUser, 4) AS UNSIGNED), 0) + 1, 3, '0')) FROM user";
70         PreparedStatement stmt = conn.prepareStatement(sql);
71         ResultSet rs = stmt.executeQuery();
72
73         String newId = "";
74         if(rs.next()) {
75             newId = rs.getString(1);
76         }
77
78         rs.close();
79         stmt.close();
80         return newId;
81     }
82
83     public static void updateDokter(Dokter d) {

```

Penjelasan DokterDAO.java

a. Deskripsi Umum

'DokterDAO' mengelola akses data untuk tabel 'dokter'.

b. Fungsionalitas Utama

- Menyediakan operasi CRUD lengkap untuk data dokter.
- Memiliki fungsionalitas untuk membuat ID User baru secara otomatis.

c. Komponen Penting (Code Breakdown)

- 'getAllDokter()', 'updateDokter(Dokter d)', 'deleteDokter(String idDokter)': Metode-metode ini mengikuti pola standar CRUD seperti pada 'PasienDAO'.
- 'insertDokter(Dokter d)': Metode ini memiliki logika tambahan. Sebelum menyisipkan data dokter, ia memeriksa apakah 'idUser' pada objek 'Dokter' kosong. Jika ya, ia akan memanggil 'generateNewUserId()' untuk membuat ID baru, lalu menyisipkannya ke tabel 'dokter'.
- 'generateNewUserId()': Metode helper privat ini membuat ID User baru yang unik dengan format 'USR' diikuti tiga digit angka (misal, 'USR001'). Ia melakukannya dengan mengambil nilai maksimum dari ID yang sudah ada di tabel 'user', menambahkannya dengan 1, dan memformatnya dengan leading zeros.

B. JadwalDokterDAO.java

```
5  package dao;
6
7  import model.JadwalDokter;
8  import klinik_pbo.koneksi;
9  import java.sql.*;
10 import java.util.*;
11 import javax.swing.*;
12
13 public class JadwalDokterDAO {
14     private Connection conn;
15
16     public JadwalDokterDAO() {
17         this.conn = koneksi.getKoneksi(); // Pastikan class koneksi sudah ada
18     }
19
20
21     public static List<JadwalDokter> getAllJadwalDokterLengkap() {
22         List<JadwalDokter> list = new ArrayList<>();
23         Connection conn = koneksi.getKoneksi();
24
25         try {
26             String sql = """
27                 SELECT
28                     jd.idJadwalDokter,
29                     d.nama AS namaDokter,
30                     j.hari,
31                     r.namaRuangan
32                 FROM jadwaldokter jd
33                 JOIN dokter d ON jd.idDokter = d.idDokter
34                 JOIN jadwal j ON jd.idJadwal = j.idJadwal
35                 JOIN ruangan r ON jd.idRuangan = r.idRuangan
36             """;
37
38             PreparedStatement stmt = conn.prepareStatement(sql);
39             ResultSet rs = stmt.executeQuery();
40
41             while (rs.next()) {
42                 JadwalDokter jd = new JadwalDokter(
43                     rs.getString("idJadwalDokter"),
44                     rs.getString("namaDokter"),
45                     rs.getString("hari"),
46                     rs.getString("namaRuangan")
47                 );
48
49                 list.add(jd);
50             }
51
52             rs.close();
53             stmt.close();
54         } catch (SQLException e) {
55             e.printStackTrace();
56         }
57
58         return list;
59     }
60
61     // Method untuk insert jadwal dokter baru
62     public void insertJadwalDokter(String idDokter, String hari, String jamMulai,
63                                     String jamBerakhir, String idRuangan) throws SQLException {
64         // Insert ke tabel jadwal dulu
65         String insertJadwal = "INSERT INTO jadwal (idJadwal, hari, jamMulai, jamBerakhir) VALUES (?, ?, ?, ?)";
66         String idJadwal = "JAD" + System.currentTimeMillis(); // Generate unique ID
67
68         try (PreparedStatement ps = conn.prepareStatement(insertJadwal)) {
69             ps.setString(1, idJadwal);
70             ps.setString(2, hari);
71             ps.setString(3, jamMulai);
72             ps.setString(4, jamBerakhir);
73             ps.executeUpdate();
74         }
75
76         // Insert ke tabel jadwaldokter
77         String insertJadwalDokter = "INSERT INTO jadwaldokter (idJadwalDokter, idDokter, idJadwal, idRuangan) VALUES (?, ?, ?, ?)";
78         String idJadwalDokter = "JDK" + System.currentTimeMillis(); // Generate unique ID
79
80         try (PreparedStatement ps = conn.prepareStatement(insertJadwalDokter)) {
81             ps.setString(1, idJadwalDokter);
82             ps.setString(2, idDokter);
83             ps.setString(3, idJadwal);
84             ps.setString(4, idRuangan);
85             ps.executeUpdate();
86         }
87
88         // Method untuk mendapatkan semua data jadwal dokter (untuk refresh tabel)
89         public List<JadwalDokter> getAllJadwalDokter() throws SQLException {
90             List<JadwalDokter> list = new ArrayList<>();
```

```

91     String query = "SELECT jd.idJadwalDokter, d.nama AS namaDokter, j.hari, r.namaRuangan " +
92         "FROM jadwaldokter jd " +
93         "JOIN dokter d ON jd.idDokter = d.idDokter " +
94         "JOIN jadwal j ON jd.idJadwal = j.idJadwal " +
95         "JOIN ruangan r ON jd.idRuangan = r.idRuangan";
96
97     try (Statement stmt = conn.createStatement());
98     {
99         ResultSet rs = stmt.executeQuery(query);
100
101         while (rs.next()) {
102             list.add(new JadwalDokter(
103                 rs.getString("idJadwalDokter"),
104                 rs.getString("namaDokter"),
105                 rs.getString("hari"),
106                 rs.getString("namaRuangan")
107             ));
108         }
109     }
110     return list;
111 }
112
113 // Ambil semua jadwal dokter untuk ditampilkan di combo box
114 public Map<String, String> getAllJadwalDokterForCombo() throws SQLException {
115     Map<String, String> jadwalMap = new HashMap<>();
116     String query = "SELECT jd.idJadwalDokter, d.nama AS namaDokter, j.hari, r.namaRuangan " +
117         "FROM jadwaldokter jd " +
118         "JOIN dokter d ON jd.idDokter = d.idDokter " +
119         "JOIN jadwal j ON jd.idJadwal = j.idJadwal " +
120         "JOIN ruangan r ON jd.idRuangan = r.idRuangan";
121
122     try (Statement stmt = conn.createStatement());
123     {
124         ResultSet rs = stmt.executeQuery(query);
125
126         while (rs.next()) {
127             String id = rs.getString("idJadwalDokter");
128             String displayText = rs.getString("namaDokter") + " - " +
129                         rs.getString("hari") + " - " +
130                         rs.getString("namaRuangan");
131             jadwalMap.put(displayText, id);
132         }
133     }
134     return jadwalMap;
135 }
136
137 // Ambil detail jadwal dokter by ID
138 public Map<String, String> getJadwalDokterById(String idJadwalDokter) throws SQLException {
139     Map<String, String> data = new HashMap<>();
140     String query = "SELECT jd.*, d.nama AS namaDokter, j.hari, j.jamMulai, j.jamBerakhir, r.namaRuangan " +
141         "FROM jadwaldokter jd " +
142         "JOIN dokter d ON jd.idDokter = d.idDokter " +
143         "JOIN jadwal j ON jd.idJadwal = j.idJadwal " +
144         "JOIN ruangan r ON jd.idRuangan = r.idRuangan " +
145         "WHERE jd.idJadwalDokter = ?";
146
147     try (PreparedStatement ps = conn.prepareStatement(query)) {
148         ps.setString(1, idJadwalDokter);
149         ResultSet rs = ps.executeQuery();
150
151         if (rs.next()) {
152             data.put("idJadwalDokter", rs.getString("idJadwalDokter"));
153             data.put("idDokter", rs.getString("idDokter"));
154             data.put("namaDokter", rs.getString("namaDokter"));
155             data.put("idJadwal", rs.getString("idJadwal"));
156             data.put("hari", rs.getString("hari"));
157             data.put("jamMulai", rs.getString("jamMulai"));
158             data.put("jamBerakhir", rs.getString("jamBerakhir"));
159             data.put("idRuangan", rs.getString("idRuangan"));
160             data.put("namaRuangan", rs.getString("namaRuangan"));
161         }
162     }
163     return data;
164 }
165
166 // Update jadwal dokter
167 public void updateJadwalDokter(String idJadwalDokter, String idDokter, String hari,
168     String jamMulai, String jamBerakhir, String idRuangan) throws SQLException {
169     // Pertama dapatkan idJadwal dari jadwaldokter
170     String idJadwal = null;
171     String queryGetJadwal = "SELECT idJadwal FROM jadwaldokter WHERE idJadwalDokter = ?";
172     try (PreparedStatement ps = conn.prepareStatement(queryGetJadwal)) {
173         ps.setString(1, idJadwalDokter);
174         ResultSet rs = ps.executeQuery();
175         if (rs.next()) {
176             idJadwal = rs.getString("idJadwal");
177         }
178     }
179 }
```

```

178     if (idJadwal != null) {
179         // Update tabel jadwal
180         String queryUpdateJadwal = "UPDATE jadwal SET hari=?, jamMulai=?, jamBerakhir=? WHERE idJadwal=?";
181         try (PreparedStatement ps = conn.prepareStatement(queryUpdateJadwal)) {
182             ps.setString(1, hari);
183             ps.setString(2, jamMulai);
184             ps.setString(3, jamBerakhir);
185             ps.setString(4, idJadwal);
186             ps.executeUpdate();
187         }
188
189         // Update tabel jadwaldokter
190         String queryUpdateJadwalDokter = "UPDATE jadwaldokter SET idDokter=?, idRuangan=? WHERE idJadwalDokter=?";
191         try (PreparedStatement ps = conn.prepareStatement(queryUpdateJadwalDokter)) {
192             ps.setString(1, idDokter);
193             ps.setString(2, idRuangan);
194             ps.setString(3, idJadwalDokter);
195             ps.executeUpdate();
196         }
197     }
198
199
200     // Hapus jadwal dokter
201     public void deleteJadwalDokter(String idJadwalDokter) throws SQLException {
202         // Pertama dapatkan idJadwal untuk dihapus dari tabel jadwal
203         String idJadwal = null;
204         String queryGetJadwal = "SELECT idJadwal FROM jadwaldokter WHERE idJadwalDokter = ?";
205         try (PreparedStatement ps = conn.prepareStatement(queryGetJadwal)) {
206             ps.setString(1, idJadwalDokter);
207             ResultSet rs = ps.executeQuery();
208             if (rs.next()) {
209                 idJadwal = rs.getString("idJadwal");
210             }
211         }
212
213         if (idJadwal != null) {
214             // Hapus dari jadwaldokter terlebih dahulu
215             String queryDeleteJadwalDokter = "DELETE FROM jadwaldokter WHERE idJadwalDokter = ?";
216             try (PreparedStatement ps = conn.prepareStatement(queryDeleteJadwalDokter)) {
217                 ps.setString(1, idJadwalDokter);
218                 ps.executeUpdate();
219             }
220
221             String queryDeleteJadwal = "DELETE FROM jadwal WHERE idJadwal = ?";
222             try (PreparedStatement ps = conn.prepareStatement(queryDeleteJadwal)) {
223                 ps.setString(1, idJadwal);
224                 ps.executeUpdate();
225             }
226         }
227     }
228
229
230     // Ambil daftar dokter untuk combo box
231     public Map<String, String> getAllDokterForCombo() throws SQLException {
232         Map<String, String> dokterMap = new HashMap<>();
233         String query = "SELECT idDokter, nama FROM dokter";
234
235         try (Statement stmt = conn.createStatement());
236             ResultSet rs = stmt.executeQuery(query)) {
237
238             while (rs.next()) {
239                 dokterMap.put(rs.getString("nama"), rs.getString("idDokter"));
240             }
241         }
242         return dokterMap;
243     }
244
245
246     // Ambil daftar ruangan untuk combo box
247     public Map<String, String> getAllRuanganForCombo() throws SQLException {
248         Map<String, String> ruanganMap = new HashMap<>();
249         String query = "SELECT idRuangan, namaRuangan FROM ruangan";
250
251         try (Statement stmt = conn.createStatement());
252             ResultSet rs = stmt.executeQuery(query)) {
253
254             while (rs.next()) {
255                 ruanganMap.put(rs.getString("namaRuangan"), rs.getString("idRuangan"));
256             }
257         }
258         return ruanganMap;
259     }
260
261     public void insertJadwalDokter(String idJadwal, String idDokter, String hari,
262                                     String jamMulai, String jamBerakhir, String idRuangan) throws SQLException {
263         // Insert ke tabel jadwal
264         String insertJadwal = "INSERT INTO jadwal (idJadwal, hari, jamMulai, jamBerakhir) VALUES (?, ?, ?, ?)";

```

```

:64
:65     try (PreparedStatement ps = conn.prepareStatement(insertJadwal)) {
:66         ps.setString(1, idJadwal);
:67         ps.setString(2, hari);
:68         ps.setString(3, jamMulai);
:69         ps.setString(4, jamBerakhir);
:70         ps.executeUpdate();
:71     }
:72
:73     // Insert ke tabel jadwaldokter
:74     String insertJadwalDokter = "INSERT INTO jadwaldokter (idJadwalDokter, idDokter, idJadwal, idRuang) VALUES (?, ?, ?, ?)";
:75     String idJadwalDokter = idJadwal; // Generate ID jadwal dokter
:76
:77     try (PreparedStatement ps = conn.prepareStatement(insertJadwalDokter)) {
:78         ps.setString(1, idJadwalDokter);
:79         ps.setString(2, idDokter);
:80         ps.setString(3, idJadwal);
:81         ps.setString(4, idRuang);
:82         ps.executeUpdate();
:83     }
:84 }
:85

```

Penjelasan JadwalDokterDAO.java

a. Deskripsi Umum

'JadwalDokterDAO' adalah kelas akses data yang kompleks karena berinteraksi dengan beberapa tabel ('jadwaldokter', 'dokter', 'jadwal', 'ruangan') untuk mengelola jadwal spesifik setiap dokter.

b. Fungsionalitas Utama

- Read (Lengkap): Mengambil data jadwal yang sudah digabungkan (di-JOIN) dari beberapa tabel untuk ditampilkan secara informatif.
- Create, Update, Delete: Menangani operasi CRUD pada jadwal dokter, yang melibatkan manipulasi pada dua tabel ('jadwal' dan 'jadwaldokter').
- Utility: Menyediakan data untuk 'JComboBox' di antarmuka pengguna.

c. Komponen Penting (Code Breakdown)

- `getAllJadwalDokterLengkap()`: Menggunakan query 'SELECT' dengan 'JOIN' antar tabel 'jadwaldokter', 'dokter', 'jadwal', dan 'ruangan' untuk menghasilkan data yang siap tampil, seperti nama dokter dan nama ruangan, bukan hanya ID-nya.
- `insertJadwalDokter(...)`: Operasi insert dilakukan dalam dua langkah: pertama, menyisipkan record baru ke tabel 'jadwal'; kedua, menyisipkan record baru ke tabel 'jadwaldokter' yang merujuk ke ID jadwal yang baru saja dibuat.
- `updateJadwalDokter(...)`: Mirip dengan insert, update juga dilakukan dalam dua langkah: memperbarui tabel 'jadwal' dan kemudian memperbarui tabel 'jadwaldokter'.
- `deleteJadwalDokter(...)`: Proses penghapusan juga harus dilakukan dalam dua langkah untuk menjaga integritas data: pertama, menghapus record dari 'jadwaldokter' (tabel anak), kemudian menghapus record dari 'jadwal' (tabel induk).

- `...ForCombo()` methods: Metode seperti `getAllDokterForCombo()` dan `getAllRuanganForCombo()` menjalankan query `SELECT` untuk mengambil ID dan nama, lalu mengembalikannya sebagai `Map<String, String>`. Ini sangat berguna untuk mengisi `JComboBox` di GUI, di mana pengguna melihat nama tetapi program menggunakan ID.

C. JadwalPraktikDAO.java

```

5   package dao;
6
7   import java.sql.*;
8   import java.util.*;
9   import model.JadwalPraktik;
10
11  public class JadwalPraktikDAO {
12      private Connection conn;
13
14      public JadwalPraktikDAO(Connection conn) {
15          this.conn = conn;
16      }
17
18      public List<JadwalPraktik> getAllJadwal() throws SQLException {
19          List<JadwalPraktik> list = new ArrayList<>();
20          String query = """
21              SELECT j.idJadwal, j.hari, j.jamMulai, j.jamBerakhir
22              FROM jadwal j
23          """;
24
25          try (Statement stmt = conn.createStatement(); ResultSet rs = stmt.executeQuery(query)) {
26              while (rs.next()) {
27                  list.add(new JadwalPraktik(
28                      rs.getString("idJadwal"),
29                      rs.getString("hari"),
30                      rs.getString("jamMulai"),
31                      rs.getString("jamBerakhir")
32                  ));
33              }
34          }
35          return list;
36      }
37  }

```

Penjelasan JadwalPraktikDAO.java

d. Deskripsi Umum

`JadwalPraktikDAO` adalah DAO yang lebih sederhana, bertugas mengambil data jadwal praktik secara umum dari tabel `jadwal`.

e. Fungsionalitas Utama

Read: Mengambil semua data jadwal (ID, hari, jam mulai, jam berakhir).

f. Komponen Penting (Code Breakdown)

`'getAllJadwal()'`: Metode ini menjalankan query `SELECT` pada tabel `jadwal`. Tujuannya adalah untuk menyediakan data jadwal yang akan ditampilkan secara umum, misalnya di halaman dashboard, tanpa detail spesifik tentang dokter atau ruangan. Ini adalah DAO yang bersifat read-only.

D. ObatDAO.java

```
5   package dao;
6
7   import model.Obat;
8   import klinik_pbo.koneksi;
9   import java.sql.*;
10  import java.util.*;
11
12 public class ObatDAO {
13     public static List<Obat> getAllObat() {
14         List<Obat> list = new ArrayList<>();
15         Connection conn = koneksi.getKoneksi();
16
17         try {
18             String sql = "SELECT * FROM obat";
19             PreparedStatement stmt = conn.prepareStatement(sql);
20             ResultSet rs = stmt.executeQuery();
21
22             while (rs.next()) {
23                 Obat o = new Obat(
24                     rs.getString("idObat"),
25                     rs.getString("namaObat"),
26                     rs.getInt("stok"),
27                     rs.getDouble("harga")
28                 );
29                 list.add(o);
30             }
31
32             rs.close();
33             stmt.close();
34         } catch (SQLException e) {
35             e.printStackTrace();
36         }
37
38         return list;
39     }
40
41     public static void insertObat(Obat o) {
42         Connection conn = koneksi.getKoneksi();
43         String sql = "INSERT INTO obat (idObat, namaObat, stok, harga) VALUES (?, ?, ?, ?)";
44
45         try {
46             PreparedStatement stmt = conn.prepareStatement(sql);
47
48             PreparedStatement stmt = conn.prepareStatement(sql);
49             stmt.setString(1, o.getIdObat());
50             stmt.setString(2, o.getNamaObat());
51             stmt.setInt(3, o.getStok());
52             stmt.setDouble(4, o.getHarga());
53             stmt.executeUpdate();
54             stmt.close();
55         } catch (SQLException e) {
56             e.printStackTrace();
57         }
58
59     public static void updateObat(Obat o) {
60         Connection conn = koneksi.getKoneksi();
61         String sql = "UPDATE obat SET namaObat=?, stok=?, harga=? WHERE idObat=?";
62
63         try {
64             PreparedStatement stmt = conn.prepareStatement(sql);
65             stmt.setString(1, o.getNamaObat());
66             stmt.setInt(2, o.getStok());
67             stmt.setDouble(3, o.getHarga());
68             stmt.setString(4, o.getIdObat());
69             stmt.executeUpdate();
70             stmt.close();
71         } catch (SQLException e) {
72             e.printStackTrace();
73         }
74
75     public static void deleteObat(String idObat) {
76         Connection conn = koneksi.getKoneksi();
77         String sql = "DELETE FROM obat WHERE idObat=?";
78
79         try {
80             PreparedStatement stmt = conn.prepareStatement(sql);
81             stmt.setString(1, idObat);
82             stmt.executeUpdate();
83             stmt.close();
84         } catch (SQLException e) {
85             e.printStackTrace();
86         }
87     }
}
```

Penjelasan ObatDAO.java

a. Deskripsi Umum

'ObatDAO' adalah kelas akses data untuk tabel 'obat', yang menyimpan informasi tentang inventaris obat.

b. Fungsionalitas Utama

Menyediakan operasi CRUD lengkap untuk data obat (ID, nama, stok, harga).

c. Komponen Penting (Code Breakdown)

'getAllObat()', 'insertObat(Obat o)', 'updateObat(Obat o)', 'deleteObat(String idObat)': Keempat metode ini adalah implementasi standar dari operasi 'SELECT *', 'INSERT', 'UPDATE', dan 'DELETE' untuk tabel 'obat' menggunakan 'PreparedStatement' untuk keamanan dan efisiensi.

E. UserDAO.java

```
1  /*
2  * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
3  * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
4  */
5 package dao;
6
7 import java.sql.*;
8 import java.util.*;
9 import model.User;
10 import klinik_pbo.koneksi;
11
12 public class UserDAO {
13
14     // Mengambil semua data user dari tabel 'user'
15     public static List<User> getAllUsers() {
16         List<User> userList = new ArrayList<>();
17         String sql = "SELECT * FROM user";
18
19         try (Connection conn = koneksi.getKoneksi()) {
20             PreparedStatement stmt = conn.prepareStatement(sql);
21             ResultSet rs = stmt.executeQuery();
22
23             while (rs.next()) {
24                 String idUser = rs.getString("idUser");
25                 String username = rs.getString("username");
26                 String password = rs.getString("password");
27                 String role = rs.getString("role");
28
29                 // Gunakan username sebagai name default
30                 User user = new User(idUser, username, username, password, role);
31                 userList.add(user);
32             }
33         } catch (SQLException e) {
34             e.printStackTrace();
35         }
36
37         return userList;
38     }
39
40     // Tambahkan user baru ke database
41     public static void insertUser(User user) {
42         String sql = "INSERT INTO user (idUser, username, password, role) VALUES (?, ?, ?, ?)";
43
44         try (Connection conn = koneksi.getKoneksi()) {
45             PreparedStatement stmt = conn.prepareStatement(sql);
46
47             stmt.setString(1, user.getIdUser());
48             stmt.setString(2, user.getUsername());
49             stmt.setString(3, user.getPassword());
50             stmt.setString(4, user.getRole());
51             stmt.executeUpdate();
52
53         } catch (SQLException e) {
54             e.printStackTrace();
55         }
56     }
57
58     // (Opsional) Cek apakah username sudah digunakan
59     public static boolean isUsernameExist(String username) {
60         String sql = "SELECT COUNT(*) FROM user WHERE username = ?";
61         try (Connection conn = koneksi.getKoneksi()) {
62             PreparedStatement stmt = conn.prepareStatement(sql);
63
64             stmt.setString(1, username);
65             ResultSet rs = stmt.executeQuery();
66         }
```

```
67
68     if (rs.next()) {
69         return rs.getInt(1) > 0;
70     }
71
72 } catch (SQLException e) {
73     e.printStackTrace();
74 }
75
76     return false;
77 }
78 }
```

Penjelasan UserDAO.java

a. Deskripsi Umum

UserDAO adalah kelas yang mengelola semua interaksi dengan tabel user di database. Tabel ini menyimpan informasi kredensial (username, password) dan peran (role) untuk autentikasi.

b. Fungsionalitas Utama

- Read: Mengambil semua data pengguna dari database.
- Create: Menambahkan pengguna baru ke database.
- Utility: Memeriksa apakah sebuah username sudah ada di database.

c. Komponen Penting (Code Breakdown)

- `getAllUsers()`: Menjalankan query `SELECT * FROM user` untuk mengambil semua record pengguna. Setiap baris hasil query diubah menjadi objek User dan ditambahkan ke dalam sebuah List.
- `insertUser(User user)`: Mengeksekusi query `INSERT` untuk menyimpan objek User baru ke dalam tabel. Metode ini menggunakan PreparedStatement untuk memasukkan data dengan aman.
- `isUsernameExist(String username)`: Menjalankan query `SELECT COUNT(*)` untuk memeriksa keberadaan username. Mengembalikan true jika count > 0, yang menandakan username sudah digunakan.

F. PasienDAO.java

```
1  package dao;
2
3  import model.Pasien;
4  import klinik_pbo.koneksi;
5  import java.sql.*;
6  import java.util.*;
7
8  public class PasienDAO {
9
10     // READ
11     public static List<Pasien> getAllPasien() {
12         List<Pasien> list = new ArrayList<>();
13         Connection conn = koneksi.getKoneksi();
14
15         try {
16             String sql = "SELECT * FROM pasien";
17             PreparedStatement stmt = conn.prepareStatement(sql);
18             ResultSet rs = stmt.executeQuery();
19
20             while (rs.next()) {
21                 Pasien p = new Pasien(
22                     rs.getString("idPasien"),
23                     rs.getString("nama"),
24                     rs.getString("alamat"),
25                     rs.getString("telepon")
26                 );
27                 list.add(p);
28             }
29
30             rs.close();
31             stmt.close();
32         } catch (SQLException e) {
33             e.printStackTrace();
34         }
35
36         return list;
37     }
38
39     // CREATE
40     public static void insertPasien(Pasien p) {
41         Connection conn = koneksi.getKoneksi();
42         String sql = "INSERT INTO pasien (idPasien, nama, alamat, telepon) VALUES (?, ?, ?, ?)";
43
44         try {
45             PreparedStatement stmt = conn.prepareStatement(sql);
46             stmt.setString(1, p.getIdPasien());
47             stmt.setString(2, p.getNama());
48             stmt.setString(3, p.getAlamat());
49             stmt.setString(4, p.getTelepon());
50             stmt.executeUpdate();
51             stmt.close();
52         } catch (SQLException e) {
53             e.printStackTrace();
54         }
55     }
56
57
58     // UPDATE
59     public static void updatePasien(Pasien p) {
60         Connection conn = koneksi.getKoneksi();
61         String sql = "UPDATE pasien SET nama=?, alamat=?, telepon=? WHERE idPasien=?";
62
63         try {
64             PreparedStatement stmt = conn.prepareStatement(sql);
65             stmt.setString(1, p.getNama());
66             stmt.setString(2, p.getAlamat());
67             stmt.setString(3, p.getTelepon());
68             stmt.setString(4, p.getIdPasien());
69             stmt.executeUpdate();
70             stmt.close();
71         } catch (SQLException e) {
72             e.printStackTrace();
73         }
74     }
75
76     // DELETE
77     public static void deletePasien(String idPasien) {
78         Connection conn = koneksi.getKoneksi();
79         String sql = "DELETE FROM pasien WHERE idPasien=?";
80
81         try {
82             PreparedStatement stmt = conn.prepareStatement(sql);
83             stmt.setString(1, idPasien);
84             stmt.executeUpdate();
85             stmt.close();
86         } catch (SQLException e) {
87             e.printStackTrace();
88         }
89     }
90
91
92 }
```

Penjelasan PasienDAO.java

a. Deskripsi Umum

'PasienDAO' bertanggung jawab untuk semua operasi CRUD yang berhubungan dengan data pasien pada tabel 'pasien'.

b. Fungsionalitas Utama

- Read: Mengambil semua data pasien.
- Create: Menyisipkan data pasien baru.
- Update: Memperbarui data pasien yang sudah ada.
- Delete: Menghapus data pasien berdasarkan ID.

c. Komponen Penting (Code Breakdown)

- 'getAllPasien()': Menjalankan 'SELECT * FROM pasien', mengubah setiap baris 'ResultSet' menjadi objek 'Pasien', dan mengembalikannya sebagai 'List<Pasien>'.
- 'insertPasien(Pasien p)': Menggunakan 'PreparedStatement' untuk mengeksekusi query 'INSERT INTO pasien' dengan data dari objek 'Pasien' yang diberikan.
- 'updatePasien(Pasien p)': Mengeksekusi query 'UPDATE pasien' untuk mengubah data record yang cocok dengan 'idPasien' dari objek 'Pasien' yang diberikan.
- 'deletePasien(String idPasien)': Mengeksekusi query 'DELETE FROM pasien' di mana 'idPasien' cocok dengan parameter yang diberikan.

G. ResepPasienDAO.java

```
1 package dao;
2
3 import model.ResepPasien;
4 import klinik_pbo.koneksi;
5 import java.sql.SQLException;
6 import java.sql.*;
7 import java.util.*;
8 import java.sql.Date;
9
10
11 public class ResepPasienDAO {
12     private Connection conn;
13
14     public ResepPasienDAO () {
15         this.conn = koneksi.getKoneksi();
16     }
17
18     // Ambil semua data resep lengkap
19     public static List<ResepPasien> getAllResepLengkap () {
20         List<ResepPasien> list = new ArrayList<>();
21         Connection conn = koneksi.getKoneksi();
22
23         try {
24             String sql = """
25                 SELECT r.idResep, r.penyakit, r.tanggal, r.status,
26                 p.name AS namaPasien,
27                 d.name AS namaDokter,
28                 o.namaObat
29
30                 FROM resep r
31                 JOIN pasien p ON r.idPasien = p.idPasien
32                 JOIN dokter d ON r.idDokter = d.idDokter
33                 LEFT JOIN obat o ON r.idObat = o.idObat
34             """;
35
36             PreparedStatement stmt = conn.prepareStatement(sql);
37             ResultSet rs = stmt.executeQuery();
38
39             while (rs.next()) {
40                 ResepPasien resep = new ResepPasien(
41                     rs.getString("idResep"),
42                     rs.getString("penyakit"),
43                     rs.getDate("tanggal"),
44                     rs.getString("namaPasien"),
45                     rs.getString("namaDokter"),
46                     rs.getString("namaObat")
47                 );
48
49                 list.add(resep);
50             }
51         } catch (SQLException e) {
52             e.printStackTrace();
53         }
54     }
55 }
```

```

45     rs.getString("namaDokter"),
46     rs.getString("status")
47   );
48   list.add(resep);
49 }
50
51 rs.close();
52 stmt.close();
53 catch (SQLException e) {
54   e.printStackTrace();
55 }
56
57 return list;
58 }
59
60 // Insert resep
61 public void insertResep(String idResep, String penyakit, Date tanggal, String idPasien, String idDokter, String idObat, String status) throws SQLException {
62   String query = "INSERT INTO resep (idResep, penyakit, tanggal, idPasien, idDokter, idObat, status) VALUES (?, ?, ?, ?, ?, ?, ?)";
63   try (PreparedStatement ps = conn.prepareStatement(query)) {
64     ps.setString(1, idResep);
65     ps.setString(2, penyakit);
66     ps.setDate(3, tanggal);
67     ps.setString(4, idPasien);
68     ps.setString(5, idDokter);
69     ps.setString(6, idObat);
70     ps.setString(7, status);
71     ps.executeUpdate();
72   }
73 }
74
75
76 // Update resep
77 public void updateResep(String idResep, String penyakit, Date tanggal, String idPasien, String idDokter, String idObat, String status) throws SQLException {
78   String query = "UPDATE resep SET penyakit=?, tanggal=?, idPasien=?, idDokter=?, idObat=?, status=? WHERE idResep=?";
79   try (PreparedStatement ps = conn.prepareStatement(query)) {
80     ps.setString(1, penyakit);
81     ps.setDate(2, tanggal);
82     ps.setString(3, idPasien);
83     ps.setString(4, idDokter);
84     ps.setString(5, idObat);
85     ps.setString(6, status);
86     ps.setString(7, idResep);
87     ps.executeUpdate();
88   }
89 }
90
91 public void updateResepTanpaStatus(String idResep, String penyakit, Date tanggal, String idPasien, String idDokter, String idObat) throws SQLException {
92   String query = "UPDATE resep SET penyakit=?, tanggal=?, idPasien=?, idDokter=?, idObat=? WHERE idResep=?";
93   try (PreparedStatement ps = conn.prepareStatement(query)) {
94     ps.setString(1, penyakit);
95     ps.setDate(2, tanggal);
96     ps.setString(3, idPasien);
97     ps.setString(4, idDokter);
98     ps.setString(5, idObat);
99     ps.setString(6, idResep);
100    ps.executeUpdate();
101  }
102
103
104 public void updateStatusResep(String idResep, String status) throws SQLException {
105   String query = "UPDATE resep SET status=? WHERE idResep=?";
106   try (PreparedStatement ps = conn.prepareStatement(query)) {
107     ps.setString(1, status);
108     ps.setString(2, idResep);
109     ps.executeUpdate();
110   }
111 }
112
113
114
115 // Delete resep
116 public void deleteResep(String idResep) throws SQLException {
117   String query = "DELETE FROM resep WHERE idResep = ?";
118   try (PreparedStatement ps = conn.prepareStatement(query)) {
119     ps.setString(1, idResep);
120     ps.executeUpdate();
121   }
122 }
123
124 // Ambil semua data resep (untuk refresh tabel)
125 public List<ResepPasien> getAllResep() throws SQLException {
126   List<ResepPasien> list = new ArrayList<>();
127   String query = """
128     SELECT r.idResep, r.penyakit, r.tanggal, r.status,
129           p.nama AS namaPasien,
130           d.nama AS namaDokter,

```

```

131     :     o.namaObat
132     FROM resep r
133     JOIN pasien p ON r.idPasien = p.idPasien
134     JOIN dokter d ON r.idDokter = d.idDokter
135     LEFT JOIN obat o ON r.idObat = o.idObat
136     """;
137
138     try (Statement stmt = conn.createStatement());
139     {
140         ResultSet rs = stmt.executeQuery(query));
141         while (rs.next());
142             list.add(new ResepPasien(
143                 rs.getString("idResep"),
144                 rs.getString("penyakit"),
145                 rs.getDate("tanggal"),
146                 rs.getString("namaPasien"),
147                 rs.getString("namaDokter"),
148                 rs.getString("namaObat"),
149                 rs.getString("status")
150             ));
151         }
152
153     return list;
154 }
155
156 // Get data pasien untuk combobox
157 public Map<String, String> getAllPasienForCombo() throws SQLException {
158     Map<String, String> pasienMap = new HashMap<>();
159     String query = "SELECT idPasien, nama FROM pasien";
160     try (Statement stmt = conn.createStatement());
161     {
162         ResultSet rs = stmt.executeQuery(query);
163         while (rs.next());
164             pasienMap.put(rs.getString("nama"), rs.getString("idPasien"));
165         }
166     return pasienMap;
167 }
168
169 // Get data dokter untuk combobox
170 public Map<String, String> getAllDokterForCombo() throws SQLException {
171     Map<String, String> dokterMap = new HashMap<>();
172     String query = "SELECT idDokter, nama FROM dokter";
173     try (Statement stmt = conn.createStatement());
174     {
175         ResultSet rs = stmt.executeQuery(query);
176         while (rs.next());
177             dokterMap.put(rs.getString("nama"), rs.getString("idDokter"));
178         }
179     return dokterMap;
180 }
181
182 // Get data obat untuk combobox
183 public Map<String, String> getAllObatForCombo() throws SQLException {
184     Map<String, String> obatMap = new HashMap<>();
185     String query = "SELECT idObat, namaObat FROM obat";
186     try (Statement stmt = conn.createStatement());
187     {
188         ResultSet rs = stmt.executeQuery(query);
189         while (rs.next());
190             obatMap.put(rs.getString("namaObat"), rs.getString("idObat"));
191         }
192     return obatMap;
193 }
194
195 // Get data resep berdasarkan ID
196 public Map<String, String> getResepById(String idResep) throws SQLException {
197     Map<String, String> data = new HashMap<>();
198     String query = """
199         SELECT r.*, p.nama AS namaPasien, d.nama AS namaDokter, o.namaObat
200         FROM resep r
201         JOIN pasien p ON r.idPasien = p.idPasien
202         JOIN dokter d ON r.idDokter = d.idDokter
203         LEFT JOIN obat o ON r.idObat = o.idObat
204         WHERE r.idResep = ?
205         """;

206     try (PreparedStatement ps = conn.prepareStatement(query));
207     {
208         ps.setString(1, idResep);
209         ResultSet rs = ps.executeQuery();

210         if (rs.next());
211             data.put("idResep", rs.getString("idResep"));
212             data.put("penyakit", rs.getString("penyakit"));
213             data.put("tanggal", rs.getString("tanggal"));
214             data.put("idPasien", rs.getString("idPasien"));
215             data.put("idDokter", rs.getString("idDokter"));
216             ...
217         }
218     }
219 }

```

```

217     data.put("idObat", rs.getString("idObat"));
218     data.put("status", rs.getString("status")); // tambahkan status
219     data.put("namaPasien", rs.getString("namaPasien"));
220     data.put("namaDokter", rs.getString("namaDokter"));
221     data.put("namaObat", rs.getString("namaObat"));
222   }
223 }
224
225 return data;
226
227
228
229 }

```

Penjelasan ResepPasienDAO.java

a. Deskripsi Umum

'ResepPasienDAO' adalah DAO kompleks lainnya yang mengelola semua akses data terkait tabel 'resep' dan tabel-tabel lain yang berelasi dengannya ('pasien', 'dokter', 'obat').

b. Fungsionalitas Utama

- Read (Lengkap)**: Mengambil data resep yang sudah digabungkan dari beberapa tabel.
- CRUD: Menyediakan operasi 'insert', 'update', dan 'delete' untuk data resep.
- Update Spesifik: Menyediakan metode update yang lebih spesifik, seperti hanya mengubah status resep.
- Utility: Menyediakan data untuk 'JComboBox'.

c. Komponen Penting (Code Breakdown)

- 'getAllResepLengkap()': Menggunakan query 'SELECT' dengan 'JOIN' ke tabel 'pasien', 'dokter', dan 'obat' untuk mengumpulkan informasi lengkap tentang setiap resep dalam satu kali query.
- 'insertResep(...)': Metode standar untuk menyisipkan record resep baru.
- Metode Update Bervariasi: Terdapat beberapa metode update:
 - 'updateResep(...)': Memperbarui semua field dalam sebuah resep.
 - 'updateResepTanpaStatus(...)': Hanya memperbarui detail resep tanpa mengubah statusnya. Berguna untuk dokter yang mengedit resep.
 - 'updateStatusResep(...)': Hanya memperbarui status resep. Berguna untuk admin yang menandai resep sudah ditebus.
- '...ForCombo()' methods: 'getAllPasienForCombo()', 'getAllDokterForCombo()', dan 'getAllObatForCombo()' berfungsi untuk mengambil data yang akan digunakan untuk mengisi 'JComboBox' pada form pembuatan/pengeditan resep.
- 'getResepById(String idResep)': Mengambil semua detail dari satu resep spesifik berdasarkan ID-nya, biasanya digunakan untuk mengisi form edit dengan data yang ada.

H. RuanganDAO.java

```
5  package dao;
6
7  import model.Ruangan;
8  import klinik_pbo.koneksi;
9  import java.sql.*;
10 import java.util.*;
11
12 public class RuanganDAO {
13     public static List<Ruangan> getAllRuangan() {
14         List<Ruangan> list = new ArrayList<>();
15         Connection conn = koneksi.getKoneksi();
16
17         try {
18             String sql = "SELECT * FROM ruangan";
19             PreparedStatement stmt = conn.prepareStatement(sql);
20             ResultSet rs = stmt.executeQuery();
21
22             while (rs.next()) {
23                 Ruangan r = new Ruangan(
24                     rs.getString("idRuangan"),
25                     rs.getString("namaRuangan"),
26                     rs.getInt("lantai"),
27                     rs.getInt("kapasitas")
28                 );
29                 list.add(r);
30             }
31
32             rs.close();
33             stmt.close();
34         } catch (SQLException e) {
35             e.printStackTrace();
36         }
37
38         return list;
39     }
40
41     public static void insertRuangan(Ruangan r) {
42         Connection conn = koneksi.getKoneksi();
43         String sql = "INSERT INTO ruangan (idRuangan, namaRuangan, lantai, kapasitas) VALUES (?, ?, ?, ?)";
44
45         try {
46             PreparedStatement stmt = conn.prepareStatement(sql);
47             stmt.setString(1, r.getIdRuangan());
48             stmt.setString(2, r.getNamaRuangan());
49             stmt.setInt(3, r.getLantai());
50             stmt.setInt(4, r.getKapasitas());
51             stmt.executeUpdate();
52             stmt.close();
53         } catch (SQLException e) {
54             e.printStackTrace();
55         }
56     }
57
58     public static void updateRuangan(Ruangan r) {
59         Connection conn = koneksi.getKoneksi();
60         String sql = "UPDATE ruangan SET namaRuangan=? , lantai=? , kapasitas=? WHERE idRuangan=?";
61
62         try {
63             PreparedStatement stmt = conn.prepareStatement(sql);
64             stmt.setString(1, r.getNamaRuangan());
65             stmt.setInt(2, r.getLantai());
66             stmt.setInt(3, r.getKapasitas());
67             stmt.setString(4, r.getIdRuangan());
68             stmt.executeUpdate();
69             stmt.close();
70         } catch (SQLException e) {
71             e.printStackTrace();
72         }
73     }
74
75     public static void deleteRuangan(String idRuangan) {
76         Connection conn = koneksi.getKoneksi();
77         String sql = "DELETE FROM ruangan WHERE idRuangan=?";
78
79         try {
80             PreparedStatement stmt = conn.prepareStatement(sql);
81             stmt.setString(1, idRuangan);
82             stmt.executeUpdate();
83             stmt.close();
84         } catch (SQLException e) {
85             e.printStackTrace();
86         }
87     }
88 }
```

Penjelasan RuanganDAO.java

a. Deskripsi Umum

'RuanganDAO' mengelola semua interaksi dengan tabel 'ruangan' di database.

b. Fungsionalitas Utama

Menyediakan operasi CRUD lengkap untuk data ruangan (ID, nama, lantai, kapasitas).

c. Komponen Penting (Code Breakdown)

'getAllRuangans()', 'insertRuangans(Ruangan r)', 'updateRuangans(Ruangan r)', 'deleteRuangans(String idRuangans)***: Metode-metode ini menyediakan fungsionalitas 'SELECT *', 'INSERT', 'UPDATE', dan 'DELETE' yang standar dan aman untuk tabel 'ruangan', mirip dengan DAO lainnya.

3.2.3 Folder Klinik PBO

Laporan Penjelasan Kode Sumber Aplikasi Klinik PBO. Dokumen ini berisi penjelasan fungsionalitas dari setiap file kode sumber Java (.java) yang membangun aplikasi sistem informasi Klinik PBO. Aplikasi ini menggunakan Java Swing untuk antarmuka pengguna (GUI) dan JDBC untuk koneksi ke database MySQL.

A. DaftarDokterAdmin.java

```
5  package klinik_pbo;
6
7  import dao.DokterDAO;
8  import java.util.List;
9  import javax.swing.table.DefaultTableModel;
10 import model.Dokter;
11 import java.sql.Connection;
12 import java.sql.Statement;
13 import java.sql.ResultSet;
14 import java.sql.SQLException;
15 import java.util.HashMap;
16 import java.util.Map;
17 import javax.swing.JComboBox;
18 import javax.swing.JOptionPane;
19 import javax.swing.JTextField;
20 import java.sql.PreparedStatement;
21 import java.util.ArrayList;
22 import model.User;
23
24 /**
25 *
26 * @author avvjelly
27 */
28 public class DaftarDokterAdmin extends javax.swing.JFrame {
29     private void tampilkanDataDokter() {
30         List<User> daftar = new ArrayList<>(DokterDAO.getAllDokter());
31         DefaultTableModel model = (DefaultTableModel) TabelDaftarDokter.getModel();
32         model.setRowCount(0); // Bersihkan isi tabel
33
34         for (User u : daftar) {
35             if (u instanceof Dokter d) {
36                 Object[] row = {
37                     d.getIdDokter(),
38                     d.getNama(),
39                     d.getSpesialisasi(),
40                     d.getIdUser()
41                 };
42                 model.addRow(row);
43             }
44         }
45     }
46
47 }
```

```
53  public DaftarDokterAdmin() {
54      initComponents();
55      tampilkanDataDokter();
56  }
57
58  /**
59   * This method is called from within the constructor to initialize the form.
60   * WARNING: Do NOT modify this code. The content of this method is always
61   * regenerated by the Form Editor.
62   */
63  @SuppressWarnings("unchecked")
64  // Generated Code
65
66
67  private void ButtonDashboardActionPerformed(java.awt.event.ActionEvent evt) {
68      // TODO add your handling code here:
69      // 1. Tutup form saat ini
70      this.dispose();
71
72      // 2. Buka form
73      DashboardAdmin dashboard = new DashboardAdmin();
74      dashboard.setVisible(true);
75  }
76
77  private void ButtonDaftarPasienActionPerformed(java.awt.event.ActionEvent evt) {
78      // TODO add your handling code here:
79      // 1. Tutup form saat ini
80      this.dispose();
81
82      // 2. Buka form
83      DaftarPasienAdmin daftarpasien = new DaftarPasienAdmin();
84      daftarpasien.setVisible(true);
85  }
86
87  private void ButtonDaftarObatActionPerformed(java.awt.event.ActionEvent evt) {
88      // TODO add your handling code here:
89      // 1. Tutup form saat ini
90      this.dispose();
91
92      // 2. Buka form
93      DaftarObatAdmin daftarobat = new DaftarObatAdmin();
94      daftarobat.setVisible(true);
95  }
96
97
98  private void ButtonDaftarRuangActionPerformed(java.awt.event.ActionEvent evt) {
99      // TODO add your handling code here:
100     // 1. Tutup form saat ini
101     this.dispose();
102
103     // 2. Buka form
104     DaftarRuangAdmin daftarruangan = new DaftarRuangAdmin();
105     daftarruangan.setVisible(true);
106 }
107
108  private void ButtonPendaftaranPasienActionPerformed(java.awt.event.ActionEvent evt) {
109      // TODO add your handling code here:
110      // 1. Tutup form saat ini
111      this.dispose();
112
113      // 2. Buka form
114      PendaftaranPasienAdmin pendaftarpasien = new PendaftaranPasienAdmin();
115      pendaftarpasien.setVisible(true);
116 }
117
118  private void ButtonPendaftaranDokterActionPerformed(java.awt.event.ActionEvent evt) {
119      // TODO add your handling code here:
120      // 1. Tutup form saat ini
121      this.dispose();
122
123      // 2. Buka form
124      PendaftaranDokterAdmin daftardokter = new PendaftaranDokterAdmin();
125      daftardokter.setVisible(true);
126 }
127
128  private void ButtonPendaftaranObatActionPerformed(java.awt.event.ActionEvent evt) {
129      // TODO add your handling code here:
130      // 1. Tutup form saat ini
131      this.dispose();
132
133      // 2. Buka form
134      PendaftaranObatAdmin daftarobat = new PendaftaranObatAdmin();
135      daftarobat.setVisible(true);
136 }
```

```

197 private void ButtonJadwalDokterActionPerformed(java.awt.event.ActionEvent evt) {
198     // TODO add your handling code here:
199     // 1. Tutup form saat ini
200     this.dispose();
201
202     // 2. Buka form
203     JadwalDokterAdmin daftarjadwal = new JadwalDokterAdmin();
204     daftarjadwal.setVisible(true);
205 }
206
207 private void ButtonResepPasienActionPerformed(java.awt.event.ActionEvent evt) {
208     // TODO add your handling code here:
209     // 1. Tutup form saat ini
210     this.dispose();
211
212     // 2. Buka form
213     ResepPasienAdmin daftarrsep = new ResepPasienAdmin();
214     daftarrsep.setVisible(true);
215 }
216
217 private void ButtonLogoutActionPerformed(java.awt.event.ActionEvent evt) {
218     // TODO add your handling code here:
219     // Tampilkan dialog konfirmasi
220     int pilihan = JOptionPane.showConfirmDialog(this,
221         "Apakah Anda yakin ingin logout?",
222         "Konfirmasi Logout",
223         JOptionPane.YES_NO_OPTION);
224
225     // Jika pengguna memilih "Yes"
226     if (pilihan == JOptionPane.YES_OPTION) {
227         // 1. Tutup form saat ini
228         this.dispose();
229
230         // 2. Buka form login
231         Login login = new Login();
232         login.setVisible(true);
233     }
234 }
235
236 private void ButtonDeleteDokterActionPerformed(java.awt.event.ActionEvent evt) {
237     // TODO add your handling code here:
238     try {
239         Connection conn = koneksi.getKoneksi();
240         Statement stmt = conn.createStatement();
241         ResultSet rs = stmt.executeQuery("SELECT idDokter, nama FROM dokter");
242
243         JComboBox<String> comboBox = new JComboBox<>();
244         Map<String, String> dokterMap = new HashMap<>();
245
246         while (rs.next()) {
247             String id = rs.getString("idDokter");
248             String nama = rs.getString("nama");
249             comboBox.addItem(nama);
250             dokterMap.put(nama, id);
251         }
252
253         if (dokterMap.isEmpty()) {
254             JOptionPane.showMessageDialog(null, "Tidak ada dokter yang tersedia untuk dihapus.");
255             return;
256         }
257
258         int option = JOptionPane.showConfirmDialog(null, comboBox, "Pilih Dokter untuk Dihapus", JOptionPane.OK_CANCEL_OPTION);
259         if (option == JOptionPane.OK_OPTION) {
260             String namaDipilih = (String) comboBox.getSelectedItem();
261             String idDipilih = dokterMap.get(namaDipilih);
262
263             int konfirmasi = JOptionPane.showConfirmDialog(null, "Yakin ingin menghapus dokter " + namaDipilih + "?", "Konfirmasi", JOptionPane.YES_NO_OPTION);
264             if (konfirmasi == JOptionPane.YES_OPTION) {
265                 DokterDAO.deleteDokter(idDipilih);
266                 JOptionPane.showMessageDialog(null, "Dokter berhasil dihapus.");
267                 tampilanDataDokter(); // ini yang akan refresh tabel
268             }
269         }
270
271         stmt.close();
272         conn.close();
273     } catch (SQLException e) {
274         e.printStackTrace();
275         JOptionPane.showMessageDialog(null, "Terjadi kesalahan database: " + e.getMessage());
276     } catch (Exception e) {
277         e.printStackTrace();
278         JOptionPane.showMessageDialog(null, "Terjadi kesalahan umum: " + e.getMessage());
279     }
280 }

```

```

479     }
480   }
481
482   private void ButtonEditDokterActionPerformed(java.awt.event.ActionEvent evt) {
483     // TODO add your handling code here:
484     try {
485       Connection conn = koneksi.getKoneksi();
486       Statement stmt = conn.createStatement();
487       ResultSet rs = stmt.executeQuery("SELECT idDokter, nama FROM dokter");
488
489       JComboBox<String> comboBox = new JComboBox<>();
490       Map<String, String> dokterMap = new HashMap<>();
491
492       while (rs.next()) {
493         String id = rs.getString("idDokter");
494         String nama = rs.getString("nama");
495         comboBox.addItem(nama);
496         dokterMap.put(nama, id);
497       }
498
499       if (dokterMap.isEmpty()) {
500         JOptionPane.showMessageDialog(null, "Tidak ada dokter yang tersedia untuk dieredit.");
501         return;
502       }
503
504       int option = JOptionPane.showConfirmDialog(null, comboBox, "Pilih Dokter untuk Dieredit", JOptionPane.OK_CANCEL_OPTION);
505       if (option == JOptionPane.OK_OPTION) {
506         String namaDipilih = (String) comboBox.getSelectedItem();
507         String idDipilih = dokterMap.get(namaDipilih);
508
509         PreparedStatement ps = conn.prepareStatement("SELECT * FROM dokter WHERE idDokter=?");
510         ps.setString(1, idDipilih);
511         ResultSet detail = ps.executeQuery();
512
513         if (detail.next()) {
514           String nama = detail.getString("nama");
515           String spesialisasi = detail.getString("spesialisasi");
516           String idUserDipilih = detail.getString("idUser");
517
518           JTextField namaField = new JTextField(nama);
519           JTextField spesialisasiField = new JTextField(spesialisasi);
520
521           Object[] fields = {
522             "Nama:", namaField,
523             "Spesialisasi:", spesialisasiField
524           };
525
526           int edit = JOptionPane.showConfirmDialog(null, fields, "Edit Data Dokter", JOptionPane.OK_CANCEL_OPTION);
527           if (edit == JOptionPane.OK_OPTION) {
528             Dokter d = new Dokter(idDipilih, namaField.getText(), spesialisasiField.getText(), idUserDipilih);
529             DokterDAO.updateDokter(d);
530             JOptionPane.showMessageDialog(null, "Data dokter berhasil diperbarui.");
531             tampilanDataDokter(); // refresh tabel
532           }
533
534         }
535
536         detail.close();
537         ps.close();
538       }
539
540       stmt.close();
541       conn.close();
542     } catch (Exception e) {
543       e.printStackTrace();
544       JOptionPane.showMessageDialog(null, "Terjadi kesalahan: " + e.getMessage());
545     }
546   }
547
548
549   /**
550    * @param args the command line arguments
551   */
552   public static void main(String args[]) {
553     /* Set the Nimbus look and feel */
554     LookAndFeelSettingCode(optional);
555
556     /* Create and display the form */
557     java.awt.EventQueue.invokeLater(new Runnable() {
558       public void run() {
559         new DaftarDokterAdmin().setVisible(true);
560       }
561     });
562   }

```

```

584 // Variables declaration - do not modify
585 private javax.swing.JButton ButtonDaftarDokter;
586 private javax.swing.JButton ButtonDaftarObat;
587 private javax.swing.JButton ButtonDaftarPasien;
588 private javax.swing.JButton ButtonDaftarRuangan;
589 private javax.swing.JButton ButtonDashboard;
590 private javax.swing.JButton ButtonDeleteDokter;
591 private javax.swing.JButton ButtonEditDokter;
592 private javax.swing.JButton ButtonJadwalDokter;
593 private javax.swing.JButton ButtonLogout;
594 private javax.swing.JButton ButtonPendaftaranDokter;
595 private javax.swing.JButton ButtonPendaftaranObat;
596 private javax.swing.JButton ButtonPendaftaranPasien;
597 private javax.swing.JButton ButtonResepPasien;
598 private javax.swing.JLabel LabelDaftarDokter;
599 private javax.swing.JPanel PanelSidebarAdmin2;
600 private javax.swing.JPanel PanelUtamaDaftarDokter;
601 private javax.swing.JTable TabelDaftarDokter;
602 private javax.swing.JScrollPane jScrollPane;
603 // End of variables declaration
604 }

```

Penjelasan DaftarDokterAdmin.java

d. Deskripsi Umum

Halaman ini digunakan oleh admin untuk melihat, mengedit (update), dan menghapus (delete) data dokter yang sudah terdaftar dalam sistem. Ini adalah implementasi dari fitur Read, Update, dan Delete (RUD) untuk entitas Dokter.

e. Fungsionalitas Utama

- Read: Menampilkan daftar semua dokter beserta spesialisasi dan ID User terkait dalam sebuah JTable.
- Update: Menyediakan tombol "Edit" yang memungkinkan admin mengubah data dokter yang dipilih.
- Delete: Menyediakan tombol "Delete" untuk menghapus data dokter dari sistem.

f. Komponen Penting (Code Breakdown)

- `tampilkanDataDokter()`: Mengambil data dari `DokterDAO.getAllDokter()` dan mempopulasikan `TabelDaftarDokter`.
- `ButtonEditDokterActionPerformed(...)`: Logika untuk tombol ini cukup kompleks:
 - Mengambil daftar nama dokter dari database untuk ditampilkan dalam sebuah JComboBox di dalam JOptionPane.
 - Setelah admin memilih dokter dan mengklik OK, data lengkap dokter tersebut diambil dari database.
 - Sebuah JOptionPane baru ditampilkan dengan JTextField yang sudah terisi data lama dokter (nama, spesialisasi).
 - Admin dapat mengubah data di field tersebut.

- Jika admin mengklik OK, objek Dokter baru dibuat dengan data yang telah diubah, lalu DokterDAO.updateDokter() dipanggil untuk menyimpan perubahan ke database.
 - Terakhir, tampilkanDataDokter() dipanggil kembali untuk me-refresh tabel.- ButtonDeleteDokterActionPerformed(...): Mirip dengan edit, admin memilih dokter dari JComboBox. Setelah konfirmasi, DokterDAO.deleteDokter() dipanggil dengan ID dokter yang dipilih. Tabel kemudian di-refresh.

B. PendaftaranDokterAdmin.java

```
5 package klinik_pbo;
6
7 import controller.DokterController;
8 import dao.UserDAO;
9 import java.util.List;
10 import javax.swing.JOptionPane;
11
12 /**
13 * @author avvjelly
14 */
15
16 public class PendaftaranDokterAdmin extends javax.swing.JFrame {
17
18     /**
19      * Creates new form PendaftaranDokterAdmin
20      */
21     public PendaftaranDokterAdmin() {
22         initComponents();
23
24         ComboBoxIDUser.removeAllItems(); // Hapus item default seperti "Item 1"
25
26         List<String> idUserList = UserDAO.getAllIdUser();
27         for (String idUser : idUserList) {
28             ComboBoxIDUser.addItem(idUser);
29         }
30     }
31
32     /**
33      * This method is called from within the constructor to initialize the form
34      * WARNING: Do NOT modify this code. The content of this method is always
35      * regenerated by the Form Editor.
36      */
37     @SuppressWarnings("unchecked")
38     Generated Code
39
40     private void ButtonDashboardActionPerformed(java.awt.event.ActionEvent evt)
41     {
42         // TODO add your handling code here:
43         // 1. Tutup form saat ini (DaftarPasienAdmin)
44         this.dispose();
45
46         // 2. Buka form DashboardAdmin
47         DashboardAdmin dashboard = new DashboardAdmin();
48         dashboard.setVisible(true);
49     }
50 }
```

```
250 private void ButtonDaftarPasienActionPerformed(java.awt.event.ActionEvent evt) {  
251     // TODO add your handling code here:  
252     // 1. Tutup form saat ini (DaftarPasienAdmin)  
253     this.dispose();  
254  
255     // 2. Buka form DaftarPasien  
256     DaftarPasienAdmin daftarpasien = new DaftarPasienAdmin();  
257     daftarpasien.setVisible(true);  
258 }  
259  
260 private void ButtonDaftarDokterActionPerformed(java.awt.event.ActionEvent evt) {  
261     // TODO add your handling code here:  
262     // 1. Tutup form saat ini (DaftarPasienAdmin)  
263     this.dispose();  
264  
265     // 2. Buka form DaftarPasien  
266     DaftarDokterAdmin daftardokter = new DaftarDokterAdmin();  
267     daftardokter.setVisible(true);  
268 }  
269  
270 private void ButtonDaftarObatActionPerformed(java.awt.event.ActionEvent evt) {  
271     // TODO add your handling code here:  
272     // 1. Tutup form saat ini  
273     this.dispose();  
274  
275     // 2. Buka form  
276     DaftarObatAdmin daftarovat = new DaftarObatAdmin();  
277     daftarovat.setVisible(true);  
278 }  
279  
280 private void ButtonDaftarRuanganActionPerformed(java.awt.event.ActionEvent evt) {  
281     // TODO add your handling code here:  
282     // 1. Tutup form saat ini  
283     this.dispose();  
284  
285     // 2. Buka form  
286     DaftarRuanganAdmin daftarruangan = new DaftarRuanganAdmin();  
287     daftarruangan.setVisible(true);  
288 }  
289  
290 private void TextfieldIDActionPerformed(java.awt.event.ActionEvent evt) {  
291     // TODO add your handling code here:  
292 }  
293  
294 private void TextfieldNamaDokterActionPerformed(java.awt.event.ActionEvent evt) {  
295     // TODO add your handling code here:  
296 }  
297  
298 private void TextfieldSpesialisasiDokterActionPerformed(java.awt.event.ActionEvent evt) {  
299     // TODO add your handling code here:  
300 }  
301  
302 private void ButtonPendaftaranPasienActionPerformed(java.awt.event.ActionEvent evt) {  
303     // TODO add your handling code here:  
304     // 1. Tutup form saat ini  
305     this.dispose();  
306  
307     // 2. Buka form  
308     PendaftaranPasienAdmin daftarpasien = new PendaftaranPasienAdmin();  
309     daftarpasien.setVisible(true);  
310 }  
311  
312 private void ButtonPendaftaranObatActionPerformed(java.awt.event.ActionEvent evt) {  
313     // TODO add your handling code here:  
314     // 1. Tutup form saat ini  
315     this.dispose();  
316  
317     // 2. Buka form  
318     PendaftaranObatAdmin daftarovat = new PendaftaranObatAdmin();  
319     daftarovat.setVisible(true);  
320 }  
321  
322 private void ButtonJadwalDokterActionPerformed(java.awt.event.ActionEvent evt) {  
323     // TODO add your handling code here:  
324     // 1. Tutup form saat ini  
325     this.dispose();  
326  
327     // 2. Buka form  
328     JadwalDokterAdmin daftarjadwal = new JadwalDokterAdmin();  
329     daftarjadwal.setVisible(true);  
330 }
```

```
423 private void ButtonSendDokterActionPerformed(java.awt.event.ActionEvent evt) {  
424     // TODO add your handling code here:  
425     try {  
426         String idDokter = TextfieldID.getText().trim();  
427         String nama = TextfieldNamaDokter.getText().trim();  
428         String spesialisasi = TextfieldSpesialisasiDokter.getText().trim();  
429  
430         // Validasi input  
431         if(idDokter.isEmpty() || nama.isEmpty() || spesialisasi.isEmpty()) {  
432             JOptionPane.showMessageDialog(this,  
433                 "Semua field harus diisi!",  
434                 "Peringatan",  
435                 JOptionPane.WARNING_MESSAGE);  
436             return;  
437         }  
438  
439         // Buat objek dokter (idUser dikosongkan, akan di-generate otomatis)  
440         Dokter dokter = new Dokter(idDokter, nama, spesialisasi, "");  
441  
442         // Simpan ke database  
443         DokterDAO.insertDokter(dokter);  
444  
445         JOptionPane.showMessageDialog(this,  
446             "Dokter berhasil didaftarkan!\nUsername: " + generateUsername(nama) +  
447             "\nPassword default: 12345",  
448             "Sukses",  
449             JOptionPane.INFORMATION_MESSAGE);  
450  
451         // Reset form  
452         TextfieldID.setText("");  
453         TextfieldNamaDokter.setText("");  
454         TextfieldSpesialisasiDokter.setText("");  
455  
456     } catch (Exception ex) {  
457         ex.printStackTrace();  
458         JOptionPane.showMessageDialog(this,  
459             "Gagal menyimpan data: " + ex.getMessage(),  
460             "Error",  
461             JOptionPane.ERROR_MESSAGE);  
462     }  
463 }
```

```
477  
478 private String generateUsername(String nama) {  
479     return nama.toLowerCase()  
480         .replaceAll("\\s+", "_")  
481         .replaceAll("[^a-zA-Z_]", "");  
482 }
```

```
473 private void ButtonLogoutActionPerformed(java.awt.event.ActionEvent evt) {  
474     // TODO add your handling code here:  
475     // Tampilkan dialog konfirmasi  
476     int pilihan = JOptionPane.showConfirmDialog(this,  
477         "Apakah Anda yakin ingin logout?",  
478         "Konfirmasi Logout",  
479         JOptionPane.YES_NO_OPTION);  
480  
481     // Jika pengguna memilih "Yes"  
482     if (pilihan == JOptionPane.YES_OPTION) {  
483         // 1. Tutup form saat ini  
484         this.dispose();  
485  
486         // 2. Buka form login  
487         Login login = new Login();  
488         login.setVisible(true);  
489     }  
490 }  
491  
492 /*  
493 * @param args the command line arguments  
494 */  
495 public static void main(String args[]) {  
496     /* Set the Nimbus look and feel */  
497     LookAndFeelSettingCode(optional);  
498  
499     /* Create and display the form */  
500     java.awt.EventQueue.invokeLater(new Runnable() {  
501         public void run() {  
502             new PendaftaranDokterAdmin().setVisible(true);  
503         }  
504     });  
505 }
```

```

527 // Variables declaration - do not modify
528 private javax.swing.JButton ButtonDaftarDokter;
529 private javax.swing.JButton ButtonDaftarObat;
530 private javax.swing.JButton ButtonDaftarPasien;
531 private javax.swing.JButton ButtonDaftarRuangan;
532 private javax.swing.JButton ButtonDashboard;
533 private javax.swing.JButton ButtonJadwalDokter;
534 private javax.swing.JButton ButtonLogout;
535 private javax.swing.JButton ButtonPendaftaranDokter;
536 private javax.swing.JButton ButtonPendaftaranObat;
537 private javax.swing.JButton ButtonPendaftaranPasien;
538 private javax.swing.JButton ButtonResepPasien;
539 private javax.swing.JButton ButtonSendDokter;
540 private javax.swing.JComboBox<String> ComboBoxIDUser;
541 private javax.swing.JLabel LabelID;
542 private javax.swing.JLabel LabelIDUser;
543 private javax.swing.JLabel LabelNamaDokter;
544 private javax.swing.JLabel LabelPendaftaranDokter;
545 private javax.swing.JLabel LabelSpesialisasiDokter;
546 private javax.swing.JPanel PanelSidebarAdmin2;
547 private javax.swing.JTextField TextfieldID;
548 private javax.swing.JTextField TextfieldNamaDokter;
549 private javax.swing.JTextField TextfieldSpesialisasiDokter;
550 private javax.swing.JPanel jPanell;
551 // End of variables declaration
552 }

```

Penjelasan PendaftaranDokterAdmin.java

a. Deskripsi Umum

Kelas ini menyediakan formulir bagi admin untuk mendaftarkan seorang dokter baru ke dalam sistem.

b. Fungsionalitas Utama

- Menyediakan field input untuk ID Dokter, Nama Dokter, dan Spesialisasi.
- Menyimpan data dokter baru ke dalam tabel dokter dan user di database.
- Secara otomatis membuatkan akun user untuk dokter yang baru didaftarkan.

c. Komponen Penting (Code Breakdown)

- **ButtonSendDokterActionPerformed(...):**
 - Mengambil data dari JTextField (ID, Nama, Spesialisasi).
 - Melakukan validasi untuk memastikan semua field terisi.
 - Membuat objek Dokter baru dengan data yang diinput.
 - Memanggil DokterDAO.insertDokter(dokter). Metode ini (di dalam DokterDAO) bertanggung jawab untuk menyisipkan data dokter baru dan secara otomatis membuatkan akun pengguna (user) yang terkait.
 - Menampilkan pesan sukses beserta username dan password default untuk dokter tersebut.
 - Mengosongkan kembali field input.

C. DaftarObatAdmin.java

```
5 package klinik_pbo;
6
7 import dao.ObatDAO;
8 import java.util.List;
9 import javax.swing.JOptionPanel;
10 import javax.swing.table.DefaultTableModel;
11 import model.Obat;
12 import java.sql.Connection;
13 import java.sql.Statement;
14 import java.sql.ResultSet;
15 import java.sql.SQLException;
16 import java.util.HashMap;
17 import java.util.Map;
18 import javax.swing.JComboBox;
19 import javax.swing.JTextField;
20 import java.sql.PreparedStatement;
21
22 /**
23 * 
24 * @author avvjelly
25 */
26 public class DaftarObatAdmin extends javax.swing.JFrame {
27     private void tampilanDataObat() {
28         List<Obat> daftar = ObatDAO.getAllObat();
29         DefaultTableModel model = (DefaultTableModel) TabelDaftarObat.getModel();
30         model.setRowCount(0); // Bersihkan isi tabel
31
32         for (Obat o : daftar) {
33             Object[] row = {
34                 o.getIdObat(),
35                 o.getNamaObat(),
36                 o.getStok(),
37                 o.getHarga()
38             };
39             model.addRow(row);
40         }
41     }
42 }
43
```

```
47 public DaftarObatAdmin() {
48     initComponents();
49     tampilanDataObat();
50 }
51
52 /**
53 * This method is called from within the constructor to initialize the form.
54 * WARNING: Do NOT modify this code. The content of this method is always
55 * regenerated by the Form Editor.
56 */
57 @SuppressWarnings("unchecked")
58 Generated Code
59
60 private void ButtonDashboardActionPerformed(java.awt.event.ActionEvent evt) {
61     // TODO add your handling code here:
62     // 1. Tutup form saat ini (DaftarPasienAdmin)
63     this.dispose();
64
65     // 2. Buka form DashboardAdmin
66     DashboardAdmin dashboard = new DashboardAdmin();
67     dashboard.setVisible(true);
68 }
69
70 private void ButtonDaftarPasienActionPerformed(java.awt.event.ActionEvent evt) {
71     // TODO add your handling code here:
72     // 1. Tutup form saat ini (DaftarPasienAdmin)
73     this.dispose();
74
75     // 2. Buka form DaftarPasien
76     DaftarPasienAdmin daftarpasien = new DaftarPasienAdmin();
77     daftarpasien.setVisible(true);
78 }
79
80 private void ButtonDaftarDokterActionPerformed(java.awt.event.ActionEvent evt) {
81     // TODO add your handling code here:
82     // 1. Tutup form saat ini (DaftarPasienAdmin)
83     this.dispose();
84
85     // 2. Buka form DaftarPasien
86     DaftarDokterAdmin daftardokter = new DaftarDokterAdmin();
87     daftardokter.setVisible(true);
88 }
```

```
351     private void ButtonDaftarRuanganActionPerformed(java.awt.event.ActionEvent evt) {  
352         // TODO add your handling code here:  
353         // 1. Tutup form saat ini  
354         this.dispose();  
355  
356         // 2. Buka form  
357         DaftarRuanganAdmin daftarruangan = new DaftarRuanganAdmin();  
358         daftarruangan.setVisible(true);  
359     }  
360  
361     private void ButtonPendaftaranPasienActionPerformed(java.awt.event.ActionEvent evt) {  
362         // TODO add your handling code here:  
363         // 1. Tutup form saat ini  
364         this.dispose();  
365  
366         // 2. Buka form  
367         PendaftaranPasienAdmin pendaftarpasien = new PendaftaranPasienAdmin();  
368         pendaftarpasien.setVisible(true);  
369     }  
370  
371     private void ButtonPendaftaranDokterActionPerformed(java.awt.event.ActionEvent evt) {  
372         // TODO add your handling code here:  
373         // 1. Tutup form saat ini  
374         this.dispose();  
375  
376         // 2. Buka form  
377         PendaftaranDokterAdmin daftardokter = new PendaftaranDokterAdmin();  
378         daftardokter.setVisible(true);  
379     }  
380  
381     private void ButtonPendaftaranObatActionPerformed(java.awt.event.ActionEvent evt) {  
382         // TODO add your handling code here:  
383         // 1. Tutup form saat ini  
384         this.dispose();  
385  
386         // 2. Buka form  
387         PendaftaranObatAdmin daftarobat = new PendaftaranObatAdmin();  
388         daftarobat.setVisible(true);  
389     }  
390  
391     private void ButtonJadwalDokterActionPerformed(java.awt.event.ActionEvent evt) {  
392         // TODO add your handling code here:  
393         // 1. Tutup form saat ini  
394         this.dispose();  
395  
396         // 2. Buka form  
397         JadwalDokterAdmin daftardjadwal = new JadwalDokterAdmin();  
398         daftardjadwal.setVisible(true);  
399     }  
400  
401     private void ButtonResepPasienActionPerformed(java.awt.event.ActionEvent evt) {  
402         // TODO add your handling code here:  
403         // 1. Tutup form saat ini  
404         this.dispose();  
405  
406         // 2. Buka form  
407         ResepPasienAdmin daftarresep = new ResepPasienAdmin();  
408         daftarresep.setVisible(true);  
409     }  
410  
411     private void ButtonLogoutActionPerformed(java.awt.event.ActionEvent evt) {  
412         // TODO add your handling code here:  
413         // Tampilkan dialog konfirmasi  
414         int pilihan = JOptionPane.showConfirmDialog(this,  
415             "Apakah Anda yakin ingin logout?",  
416             "Konfirmasi Logout",  
417             JOptionPane.YES_NO_OPTION);  
418  
419         // Jika pengguna memilih "Yes"  
420         if (pilihan == JOptionPane.YES_OPTION) {  
421             // 1. Tutup form saat ini  
422             this.dispose();  
423  
424             // 2. Buka form login  
425             Login login = new Login();  
426             login.setVisible(true);  
427         }  
428     }  
429  
430 }
```

```

30
31     private void ButtonEditObatActionPerformed(java.awt.event.ActionEvent evt) {
32         // TODO add your handling code here:
33         try {
34             Connection conn = koneksi.getKoneksi();
35             Statement stmt = conn.createStatement();
36             ResultSet rs = stmt.executeQuery("SELECT idObat, namaObat FROM obat");
37
38             JComboBox<String> comboBox = new JComboBox<String>();
39             Map<String, String> obatMap = new HashMap<String, String>();
40
41             while (rs.next()) {
42                 String id = rs.getString("idObat");
43                 String nama = rs.getString("namaObat");
44                 comboBox.addItem(nama);
45                 obatMap.put(nama, id);
46             }
47
48             if (obatMap.isEmpty()) {
49                 JOptionPane.showMessageDialog(null, "Tidak ada obat yang tersedia untuk diedit.");
50                 return;
51             }
52
53             int option = JOptionPane.showConfirmDialog(null, comboBox, "Pilih Obat untuk Diedit", JOptionPane.OK_CANCEL_OPTION);
54             if (option == JOptionPane.OK_OPTION) {
55                 String namaDipilih = (String) comboBox.getSelectedItem();
56                 String idDipilih = obatMap.get(namaDipilih);
57
58                 PreparedStatement ps = conn.prepareStatement("SELECT * FROM obat WHERE idObat=?");
59                 ps.setString(1, idDipilih);
60                 ResultSet detail = ps.executeQuery();
61
62                 if (detail.next()) {
63                     String namaObat = detail.getString("namaObat");
64                     int stok = detail.getInt("stok");
65                     double harga = detail.getDouble("harga");
66
67                     JTextField namaField = new JTextField(namaObat);
68                     JTextField stokField = new JTextField(String.valueOf(stok));
69                     JTextField hargaField = new JTextField(String.valueOf(harga));
70
71                     Object[] fields = {
72                         "Nama Obat:", namaField,
73                         "Stok:", stokField,
74                         ...
75                     };

```

```

470
471             Object[] fields = {
472                 "Nama Obat:", namaField,
473                 "Stok:", stokField,
474                 "Harga:", hargaField
475             };
476
477             int edit = JOptionPane.showConfirmDialog(null, fields, "Edit Data Obat", JOptionPane.OK_CANCEL_OPTION);
478             if (edit == JOptionPane.OK_OPTION) {
479                 Obat o = new Obat(
480                     idDipilih,
481                     namaField.getText(),
482                     Integer.parseInt(stokField.getText()),
483                     Double.parseDouble(hargaField.getText())
484                 );
485                 ObatDAO.updateObat(o);
486                 JOptionPane.showMessageDialog(null, "Data obat berhasil diperbarui.");
487                 tampilkanDataObat(); // refresh tabel jika ada
488             }
489
490             detail.close();
491             ps.close();
492         }
493
494         stmt.close();
495         conn.close();
496     } catch (Exception e) {
497         e.printStackTrace();
498         JOptionPane.showMessageDialog(null, "Terjadi kesalahan: " + e.getMessage());
499     }
500
501 }
502
503     private void ButtonDeleteObatActionPerformed(java.awt.event.ActionEvent evt) {
504         // TODO add your handling code here:
505         try {
506             Connection conn = koneksi.getKoneksi();
507             Statement stmt = conn.createStatement();
508             ResultSet rs = stmt.executeQuery("SELECT idObat, namaObat FROM obat");
509
510             JComboBox<String> comboBox = new JComboBox<String>();
511             Map<String, String> obatMap = new HashMap<String, String>();
512

```

```

503     private void ButtonDeletedObatActionPerformed(java.awt.event.ActionEvent evt) {
504         // TODO add your handling code here:
505         try {
506             Connection conn = koneksi.getKoneksi();
507             Statement stmt = conn.createStatement();
508             ResultSet rs = stmt.executeQuery("SELECT idObat, namaObat FROM obat");
509
510             JComboBox<String> comboBox = new JComboBox<>();
511             Map<String, String> obatMap = new HashMap<>();
512
513             while (rs.next()) {
514                 String id = rs.getString("idObat");
515                 String nama = rs.getString("namaObat");
516                 comboBox.addItem(name);
517                 obatMap.put(nama, id);
518             }
519
520             if (obatMap.isEmpty()) {
521                 JOptionPane.showMessageDialog(null, "Tidak ada obat yang tersedia untuk dihapus.");
522                 return;
523             }
524
525             int option = JOptionPane.showConfirmDialog(null, comboBox, "Pilih Obat untuk Dihapus", JOptionPane.OK_CANCEL_OPTION);
526             if (option == JOptionPane.OK_OPTION) {
527                 String namaDipilih = (String) comboBox.getSelectedItem();
528                 String idDipilih = obatMap.get(namaDipilih);
529
530                 int konfirmasi = JOptionPane.showConfirmDialog(null, "Yakin ingin menghapus obat '" + namaDipilih + "'?", "Konfirmasi", JOptionPane.YES_NO_OPTION);
531                 if (konfirmasi == JOptionPane.YES_OPTION) {
532                     ObatDAO.deletedObat(idDipilih);
533                     JOptionPane.showMessageDialog(null, "Obat berhasil dihapus.");
534                     tampilanDataObat(); // refresh tabel obat
535                 }
536             }
537
538             stmt.close();
539             conn.close();
540         } catch (Exception e) {
541             e.printStackTrace();
542             JOptionPane.showMessageDialog(null, "Terjadi kesalahan: " + e.getMessage());
543         }
544     }

```

```

545
546     /**
547      * @param args the command line arguments
548      */
549     public static void main(String args[]) {
550         /* Set the Nimbus look and feel */
551         Look and feel setting code (optional)
552
553         /* Create and display the form */
554         java.awt.EventQueue.invokeLater(new Runnable() {
555             public void run() {
556                 new DaftarObatAdmin().setVisible(true);
557             }
558         });
559
560         // Variables declaration - do not modify
561         private javax.swing.JButton ButtonDaftarDokter;
562         private javax.swing.JButton ButtonDaftarObat;
563         private javax.swing.JButton ButtonDaftarPasien;
564         private javax.swing.JButton ButtonDaftarRuang;
565         private javax.swing.JButton ButtonDashboard;
566         private javax.swing.JButton ButtonDeleteObat;
567         private javax.swing.JButton ButtonEditObat;
568         private javax.swing.JButton ButtonJadwalDokter;
569         private javax.swing.JButton ButtonLogout;
570         private javax.swing.JButton ButtonPendaftaranDokter;
571         private javax.swing.JButton ButtonPendaftaranObat;
572         private javax.swing.JButton ButtonPendaftaranPasien;
573         private javax.swing.JButton ButtonResepPasien;
574         private javax.swing.JLabel LabelDaftarObat;
575         private javax.swing.JPanel PanelSidebarAdmin2;
576         private javax.swing.JTable TabelDaftarObat;
577         private javax.swing.JPanel jPanel1;
578         private javax.swing.JScrollPane jScrollPane;
579         // End of variables declaration
580     }

```

Penjelasan DaftarObatAdmin.java

a. Deskripsi Umum

Kelas ini berfungsi sebagai antarmuka bagi admin untuk mengelola data inventaris obat, termasuk melihat, mengedit, dan menghapus data obat.

b. Fungsionalitas Utama

- Read: Menampilkan daftar obat beserta ID, nama, stok, dan harga dalam JTable.
- Update: Memungkinkan admin mengubah nama, stok, atau harga obat melalui tombol "Edit".
- Delete: Memungkinkan admin menghapus data obat melalui tombol "Delete".

c. Komponen Penting (Code Breakdown)

- `tampilkanDataObat()`: Mengambil semua data obat melalui `ObatDAO.getAllObat()` dan menampilkannya di TabelDaftarObat.
- `ButtonEditObatActionPerformed(...)`: Mekanismenya sangat mirip dengan `ButtonEditDokterActionPerformed`. Admin memilih obat dari JComboBox, lalu sebuah dialog muncul untuk mengedit field nama, stok, dan harga. Perubahan disimpan menggunakan `ObatDAO.updateObat()`.
- `ButtonDeleteObatActionPerformed(...)`: Admin memilih obat yang akan dihapus dari JComboBox. Setelah konfirmasi, `ObatDAO.deleteObat()` dipanggil untuk menghapus data dari database.

D. PendaftaranObatAdmin.java

```
5  package klinik_poo;
6
7  import controller.ObatController;
8  import javax.swing.JOptionPane;
9
10 // /**
11 // *
12 // * @author avvjelly
13 // */
14 public class PendaftaranObatAdmin extends javax.swing.JFrame {
15
16 // /**
17 // * Creates new form PendaftaranObatAdmin
18 // */
19 public PendaftaranObatAdmin() {
20     initComponents();
21 }
22
23 // /**
24 // * This method is called from within the constructor to initialize the form.
25 // * WARNING: Do NOT modify this code. The content of this method is always
26 // * regenerated by the Form Editor.
27 // */
28 @SuppressWarnings("unchecked")
29 // Generated Code
30
31
32 private void ButtonDashboardActionPerformed(java.awt.event.ActionEvent evt) {
33     // TODO add your handling code here:
34     // 1. Tutup form saat ini (DaftarPasienAdmin)
35     this.dispose();
36
37     // 2. Buka form DashboardAdmin
38     DashboardAdmin dashboard = new DashboardAdmin();
39     dashboard.setVisible(true);
40 }
41
42 private void ButtonDaftarPasienActionPerformed(java.awt.event.ActionEvent evt) {
43     // TODO add your handling code here:
44     // 1. Tutup form saat ini (DaftarPasienAdmin)
45     this.dispose();
46
47     // 2. Buka form DaftarPasien
48     DaftarPasienAdmin daftarpasien = new DaftarPasienAdmin();
49 }
```

```
339         daftarpasien.setVisible(true);
340     }
341
342     private void ButtonDaftarDokterActionPerformed(java.awt.event.ActionEvent evt) {
343         // TODO add your handling code here:
344         // 1. Tutup form saat ini (DaftarPasienAdmin)
345         this.dispose();
346
347         // 2. Buka form DaftarPasien
348         DaftarDokterAdmin daftardokter = new DaftarDokterAdmin();
349         daftardokter.setVisible(true);
350     }
351
352     private void ButtonDaftarObatActionPerformed(java.awt.event.ActionEvent evt) {
353         // TODO add your handling code here:
354         // 1. Tutup form saat ini
355         this.dispose();
356
357         // 2. Buka form
358         DaftarObatAdmin daftarobat = new DaftarObatAdmin();
359         daftarobat.setVisible(true);
360     }
361
362     private void ButtonDaftarRuanganActionPerformed(java.awt.event.ActionEvent evt) {
363         // TODO add your handling code here:
364         // 1. Tutup form saat ini
365         this.dispose();
366
367         // 2. Buka form
368         DaftarRuanganAdmin daftarruangan = new DaftarRuanganAdmin();
369         daftarruangan.setVisible(true);
370     }
371
372     private void TextfieldIDActionPerformed(java.awt.event.ActionEvent evt) {
373         // TODO add your handling code here:
374     }
375
376     private void ButtonPendaftaranDokterActionPerformed (java.awt.event.ActionEvent evt) {
377         // TODO add your handling code here:
378         // 1. Tutup form saat ini
379         this.dispose();
380
381         // 2. Buka form
382         PendaftaranDokterAdmin daftardokter = new PendaftaranDokterAdmin();
383         daftardokter.setVisible(true);
384     }
385
386     private void ButtonPendaftaranPasienActionPerformed (java.awt.event.ActionEvent evt) {
387         // TODO add your handling code here:
388         // 1. Tutup form saat ini
389         this.dispose();
390
391         // 2. Buka form
392         PendaftaranPasienAdmin daftarpasien = new PendaftaranPasienAdmin();
393         daftarpasien.setVisible(true);
394     }
395
396     private void ButtonJadwalDokterActionPerformed (java.awt.event.ActionEvent evt) {
397         // TODO add your handling code here:
398         // 1. Tutup form saat ini
399         this.dispose();
400
401         // 2. Buka form
402         JadwalDokterAdmin daftardjadwal = new JadwalDokterAdmin();
403         daftardjadwal.setVisible(true);
404     }
405
406     private void ButtonResepPasienActionPerformed (java.awt.event.ActionEvent evt) {
407         // TODO add your handling code here:
408         // 1. Tutup form saat ini
409         this.dispose();
410
411         // 2. Buka form
412         ResepPasienAdmin daftardesep = new ResepPasienAdmin();
413         daftardesep.setVisible(true);
414     }
```

```

416     private void ButtonSendObatActionPerformed(java.awt.event.ActionEvent evt) {
417         // TODO add your handling code here:
418         ...
419         String id = TextfieldID.getText().trim();
420         String nama = TextfieldNamaObat.getText().trim();
421         String stokText = TextfieldStokObat.getText().trim();
422         String hargaText = TextfieldHargaObat.getText().trim();
423
424         // Validasi input
425         if (id.isEmpty() || nama.isEmpty() || stokText.isEmpty() || hargaText.isEmpty()) {
426             JOptionPane.showMessageDialog(this, "Semua field harus diisi terlebih dahulu.", "Peringatan", JOptionPane.WARNING_MESSAGE);
427             return;
428         }
429
430         int stok = Integer.parseInt(stokText);
431         double harga = Double.parseDouble(hargaText);
432
433         ObatController oc = new ObatController();
434         oc.tambahObat(id, nama, stok, harga);
435
436         JOptionPane.showMessageDialog(this, "Data obat berhasil disimpan!");
437
438         // Reset form
439         TextfieldID.setText("");
440         TextfieldNamaObat.setText("");
441         TextfieldStokObat.setText("");
442         TextfieldHargaObat.setText("");
443
444     } catch (Exception e) {
445         e.printStackTrace();
446         JOptionPane.showMessageDialog(this, "Gagal menyimpan data obat.\n" + e.getMessage());
447     }
448
449 }
450
451     private void ButtonLogoutActionPerformed(java.awt.event.ActionEvent evt) {
452         // TODO add your handling code here:
453         // Tampilkan dialog konfirmasi
454         int pilihan = JOptionPane.showConfirmDialog(this,
455             "Apakah Anda yakin ingin logout?",
456             "Konfirmasi Logout",
457             JOptionPane.YES_NO_OPTION);
458
459         // Jika pengguna memilih "Yes"
460         if (pilihan == JOptionPane.YES_OPTION) {
461             // 1. Tutup form saat ini
462             this.dispose();
463
464             // 2. Buka form login
465             Login login = new Login();
466             login.setVisible(true);
467         }
468
469
470         /*
471          * @param args the command line arguments
472          */
473         public static void main(String args[]) {
474             /* Set the Nimbus look_and_feel */
475             Look_and_feel_setting_code (optional)
476
477             /* Create and display the form */
478             java.awt.EventQueue.invokeLater(new Runnable() {
479                 public void run() {
480                     new PendaftaranObatAdmin().setVisible(true);
481                 }
482             });
483
484
485         // Variables declaration - do not modify
486         private javax.swing.JButton ButtonDaftarDokter;
487         private javax.swing.JButton ButtonDaftarObat;
488         private javax.swing.JButton ButtonDaftarPasien;
489         private javax.swing.JButton ButtonDaftarRuang;
490         private javax.swing.JButton ButtonDashboard;
491         private javax.swing.JButton ButtonJadwalDokter;
492         private javax.swing.JButton ButtonLogout;
493         private javax.swing.JButton ButtonPendaftaranDokter;
494         private javax.swing.JButton ButtonPendaftaranObat;
495         private javax.swing.JButton ButtonPendaftaranPasien;
496         private javax.swing.JButton ButtonResepPasien;
497         private javax.swing.JButton ButtonSendObat;
498         private javax.swing.JLabel LabelHargaObat;
499         private javax.swing.JLabel LabelID;
500         private javax.swing.JLabel LabelNamaObat;
501         private javax.swing.JLabel LabelStokObat;
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521

```

```
520     private javax.swing.JLabel LabelNamaObat;
521     private javax.swing.JLabel LabelStokObat;
522     private javax.swing.JPanel PanelsSidebarAdmin2;
523     private javax.swing.JTextField TextfieldHargaObat;
524     private javax.swing.JTextField TextfieldID;
525     private javax.swing.JTextField TextfieldNamaObat;
526     private javax.swing.JTextField TextfieldStokObat;
527     private javax.swing.JLabel jLabel1;
528     private javax.swing.JPanel jPanel1;
529     private javax.swing.JTextField jTextField4;
530     // End of variables declaration
531 }
```

Penjelasan PendaftaranObatAdmin.java

a. Deskripsi Umum

PendaftaranObatAdmin.java adalah formulir yang digunakan oleh admin untuk menambahkan data obat baru ke dalam inventaris klinik.

b. Fungsionalitas Utama

- Menyediakan antarmuka untuk menginput ID, Nama Obat, Stok, dan Harga.
- Menyimpan data obat baru ke database melalui controller.
- Melakukan validasi input dan konversi tipe data (String ke int untuk stok dan double untuk harga).

c. Komponen Penting (Code Breakdown)

- ButtonSendObatActionPerformed(...): Logika utama saat tombol "Send" ditekan.
 - Mengambil input dari semua JTextField.
 - Memvalidasi bahwa tidak ada field yang kosong.
 - Mengonversi input stok dan harga dari String ke tipe data numerik yang sesuai (int dan double).
 - Membuat instance dari ObatController dan memanggil metode tambahObat untuk memproses penyimpanan data.
 - Menampilkan pesan sukses dan mengosongkan formulir.
- jTextField4: Terdapat sebuah JTextField dengan nama jTextField4 yang tampaknya tidak digunakan dan tidak memiliki label, ini kemungkinan adalah sisa komponen dari proses desain UI.

E. DaftarPasienAdmin.java

```
5 package klinik_pbo;
6
7 import model.Pasien;
8 import dao.PasienDAO;
9 import java.util.List;
10 import javax.swing.JOptionPane;
11 import javax.swing.table.DefaultTableModel;
12 import java.sql.Connection;
13 import java.sql.Statement;
14 import java.sql.ResultSet;
15 import java.sql.SQLException;
16 import java.util.HashMap;
17 import java.util.Map;
18 import javax.swing.JComboBox;
19 import javax.swing.JTextField;
20 import java.sql.PreparedStatement;
21
22
23
24 /**
25 * 
26 * @author avvjelly
27 */
28 public class DaftarPasienAdmin extends javax.swing.JFrame {
29     private void tampilkanDataPasien() {
30         List<Pasien> daftar = PasienDAO.getAllPasien();
31         DefaultTableModel model = (DefaultTableModel) TabelDaftarPasien.getModel();
32         model.setRowCount(0); // Bersihkan isi tabel
33
34         for (Pasien p : daftar) {
35             Object[] row = {
36                 p.getIdPasien(),
37                 p.getNama(),
38                 p.getAlamat(),
39                 p.getTelepon()
40             };
41             model.addRow(row);
42         }
43     }
44
45
46
47
48     public DaftarPasienAdmin () {
49         initComponents();
50         tampilkanDataPasien();
51     }
52
53
54 /**
55 * This method is called from within the constructor to initialize the form.
56 * WARNING: Do NOT modify this code. The content of this method is always
57 * regenerated by the Form Editor.
58 */
59 @SuppressWarnings("unchecked")
60 Generated Code
61
62     private void ButtonDashboardActionPerformed(java.awt.event.ActionEvent evt) {
63         // TODO add your handling code here:
64         // 1. Tutup form saat ini
65         this.dispose();
66
67         // 2. Buka form
68         DashboardAdmin dashboard = new DashboardAdmin();
69         dashboard.setVisible(true);
70     }
71
72     private void ButtonDaftarDokterActionPerformed(java.awt.event.ActionEvent evt) {
73         // TODO add your handling code here:
74         // 1. Tutup form saat ini
75         this.dispose();
76
77         // 2. Buka form
78         DaftarDokterAdmin daftardokter = new DaftarDokterAdmin();
79         daftardokter.setVisible(true);
80     }
81
82     private void ButtonDaftarObatActionPerformed(java.awt.event.ActionEvent evt) {
83         // TODO add your handling code here:
84         // 1. Tutup form saat ini
85         this.dispose();
86
87         // 2. Buka form
88         DaftarObatAdmin daftarobat = new DaftarObatAdmin();
89         daftarobat.setVisible(true);
90     }
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
559
560
561
562
563
564
565
566
567
568
569
569
570
571
572
573
574
575
576
577
578
579
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
598
599
599
600
601
602
603
604
605
606
607
608
609
609
610
611
612
613
614
615
616
617
617
618
619
619
620
621
622
623
623
624
625
626
627
627
628
629
629
630
631
632
633
633
634
635
635
636
637
637
638
638
639
639
640
640
641
641
642
642
643
643
644
644
645
645
646
646
647
647
648
648
649
649
650
650
651
651
652
652
653
653
654
654
655
655
656
656
657
657
658
658
659
659
660
660
661
661
662
662
663
663
664
664
665
665
666
666
667
667
668
668
669
669
670
670
671
671
672
672
673
673
674
674
675
675
676
676
677
677
678
678
679
679
680
680
681
681
682
682
683
683
684
684
685
685
686
686
687
687
688
688
689
689
690
690
691
691
692
692
693
693
694
694
695
695
696
696
697
697
698
698
699
699
700
700
701
701
702
702
703
703
704
704
705
705
706
706
707
707
708
708
709
709
710
710
711
711
712
712
713
713
714
714
715
715
716
716
717
717
718
718
719
719
720
720
721
721
722
722
723
723
724
724
725
725
726
726
727
727
728
728
729
729
730
730
731
731
732
732
733
733
734
734
735
735
736
736
737
737
738
738
739
739
740
740
741
741
742
742
743
743
744
744
745
745
746
746
747
747
748
748
749
749
750
750
751
751
752
752
753
753
754
754
755
755
756
756
757
757
758
758
759
759
760
760
761
761
762
762
763
763
764
764
765
765
766
766
767
767
768
768
769
769
770
770
771
771
772
772
773
773
774
774
775
775
776
776
777
777
778
778
779
779
780
780
781
781
782
782
783
783
784
784
785
785
786
786
787
787
788
788
789
789
790
790
791
791
792
792
793
793
794
794
795
795
796
796
797
797
798
798
799
799
800
800
801
801
802
802
803
803
804
804
805
805
806
806
807
807
808
808
809
809
810
810
811
811
812
812
813
813
814
814
815
815
816
816
817
817
818
818
819
819
820
820
821
821
822
822
823
823
824
824
825
825
826
826
827
827
828
828
829
829
830
830
831
831
832
832
833
833
834
834
835
835
836
836
837
837
838
838
839
839
840
840
841
841
842
842
843
843
844
844
845
845
846
846
847
847
848
848
849
849
850
850
851
851
852
852
853
853
854
854
855
855
856
856
857
857
858
858
859
859
860
860
861
861
862
862
863
863
864
864
865
865
866
866
867
867
868
868
869
869
870
870
871
871
872
872
873
873
874
874
875
875
876
876
877
877
878
878
879
879
880
880
881
881
882
882
883
883
884
884
885
885
886
886
887
887
888
888
889
889
890
890
891
891
892
892
893
893
894
894
895
895
896
896
897
897
898
898
899
899
900
900
901
901
902
902
903
903
904
904
905
905
906
906
907
907
908
908
909
909
910
910
911
911
912
912
913
913
914
914
915
915
916
916
917
917
918
918
919
919
920
920
921
921
922
922
923
923
924
924
925
925
926
926
927
927
928
928
929
929
930
930
931
931
932
932
933
933
934
934
935
935
936
936
937
937
938
938
939
939
940
940
941
941
942
942
943
943
944
944
945
945
946
946
947
947
948
948
949
949
950
950
951
951
952
952
953
953
954
954
955
955
956
956
957
957
958
958
959
959
960
960
961
961
962
962
963
963
964
964
965
965
966
966
967
967
968
968
969
969
970
970
971
971
972
972
973
973
974
974
975
975
976
976
977
977
978
978
979
979
980
980
981
981
982
982
983
983
984
984
985
985
986
986
987
987
988
988
989
989
990
990
991
991
992
992
993
993
994
994
995
995
996
996
997
997
998
998
999
999
1000
1000
1001
1001
1002
1002
1003
1003
1004
1004
1005
1005
1006
1006
1007
1007
1008
1008
1009
1009
1010
1010
1011
1011
1012
1012
1013
1013
1014
1014
1015
1015
1016
1016
1017
1017
1018
1018
1019
1019
1020
1020
1021
1021
1022
1022
1023
1023
1024
1024
1025
1025
1026
1026
1027
1027
1028
1028
1029
1029
1030
1030
1031
1031
1032
1032
1033
1033
1034
1034
1035
1035
1036
1036
1037
1037
1038
1038
1039
1039
1040
1040
1041
1041
1042
1042
1043
1043
1044
1044
1045
1045
1046
1046
1047
1047
1048
1048
1049
1049
1050
1050
1051
1051
1052
1052
1053
1053
1054
1054
1055
1055
1056
1056
1057
1057
1058
1058
1059
1059
1060
1060
1061
1061
1062
1062
1063
1063
1064
1064
1065
1065
1066
1066
1067
1067
1068
1068
1069
1069
1070
1070
1071
1071
1072
1072
1073
1073
1074
1074
1075
1075
1076
1076
1077
1077
1078
1078
1079
1079
1080
1080
1081
1081
1082
1082
1083
1083
1084
1084
1085
1085
1086
1086
1087
1087
1088
1088
1089
1089
1090
1090
1091
1091
1092
1092
1093
1093
1094
1094
1095
1095
1096
1096
1097
1097
1098
1098
1099
1099
1100
1100
1101
1101
1102
1102
1103
1103
1104
1104
1105
1105
1106
1106
1107
1107
1108
1108
1109
1109
1110
1110
1111
1111
1112
1112
1113
1113
1114
1114
1115
1115
1116
1116
1117
1117
1118
1118
1119
1119
1120
1120
1121
1121
1122
1122
1123
1123
1124
1124
1125
1125
1126
1126
1127
1127
1128
1128
1129
1129
1130
1130
1131
1131
1132
1132
1133
1133
1134
1134
1135
1135
1136
1136
1137
1137
1138
1138
1139
1139
1140
1140
1141
1141
1142
1142
1143
1143
1144
1144
1145
1145
1146
1146
1147
1147
1148
1148
1149
1149
1150
1150
1151
1151
1152
1152
1153
1153
1154
1154
1155
1155
1156
1156
1157
1157
1158
1158
1159
1159
1160
1160
1161
1161
1162
1162
1163
1163
1164
1164
1165
1165
1166
1166
1167
1167
1168
1168
1169
1169
1170
1170
1171
1171
1172
1172
1173
1173
1174
1174
1175
1175
1176
1176
1177
1177
1178
1178
1179
1179
1180
1180
1181
1181
1182
1182
1183
1183
1184
1184
1185
1185
1186
1186
1187
1187
1188
1188
1189
1189
1190
1190
1191
1191
1192
1192
1193
1193
1194
1194
1195
1195
1196
1196
1197
1197
1198
1198
1199
1199
1200
1200
1201
1201
1202
1202
1203
1203
1204
1204
1205
1205
1206
1206
1207
1207
1208
1208
1209
1209
1210
1210
1211
1211
1212
1212
1213
1213
1214
1214
1215
1215
1216
1216
1217
1217
1218
1218
1219
1219
1220
1220
1221
1221
1222
1222
1223
1223
1224
1224
1225
1225
1226
1226
1227
1227
1228
1228
1229
1229
1230
1230
1231
1231
1232
1232
1233
1233
1234
1234
1235
1235
1236
1236
1237
1237
1238
1238
1239
1239
1240
1240
1241
1241
1242
1242
1243
1243
1244
1244
1245
1245
1246
1246
1247
1247
1248
1248
1249
1249
1250
1250
1251
1251
1252
1252
1253
1253
1254
1254
1255
1255
1256
1256
1257
1257
1258
1258
1259
1259
1260
1260
1261
1261
1262
1262
1263
1263
1264
1264
1265
1265
1266
1266
1267
1267
1268
1268
1269
1269
1270
1270
1271
1271
1272
1272
1273
1273
1274
1274
1275
1275
1276
1276
1277
1277
1278
1278
1279
1279
1280
1280
1281
1281
1282
1282
1283
1283
1284
1284
1285
1285
1286
1286
1287
1287
1288
1288
1289
1289
1290
1290
1291
1291
1292
1292
1293
1293
1294
1294
1295
1295
1296
1296
1297
1297
1298
1298
1299
1299
1300
1300
1301
1301
1302
1302
1303
1303
1304
1304
1305
1305
1306
1306
1307
1307
1308
1308
1309
1309
1310
1310
1311
1311
1312
1312
1313
1313
1314
1314
1315
1315
1316
1316
1317
1317
1318
1318
1319
1319
1320
1320
1321
1321
1322
1322
1323
1323
1324
1324
1325
1325
1326
1326
1327
1327
1328
1328
1329
1329
1330
1330
1331
1331
1332
1332
1333
1333
1334
1334
1335
1335
1336
1336
1337
1337
1338
1338
1339
1339
1340
1340
1341
1341
1342
1342
1343
1343
1344
1344
1345
1345
1346
1346
1347
1347
1348
1348
1349
1349
1350
1350
1351
1351
1352
1352
1353
1353
1354
1354
1355
1355
1356
1356
1357
1357
1358
1358
1359
1359
1360
1360
1361
1361
1362
1362
1363
1363
1364
1364
1365
1365
1366
1366
1367
1367
1368
1368
1369
1369
1370
1370
1371
1371
1372
1372
1373
1373
1374
1374
1375
1375
1376
1376
1377
1377
1378
1378
1379
1379
1380
1380
1381
1381
1382
1382
1383
1383
1384
1384
1385
1385
1386
1386
1387
1387
1388
1388
1389
1389
1390
1390
1391
1391
1392
1392
1393
1393
1394
1394
1395
1395
1396
1396
1397
1397
1398
1398
1399
1399
1400
1400
1401
1401
1402
1402
1403
1403
1404
1404
1405
1405
1406
1406
1407
1407
1408
1408
1409
1409
1410
1410
1411
1411
1412
1412
1413
1413
1414
1414
1415
1415
1416
1416
1417
1417
1418
1418
1419
1419
1420
1420
1421
1421
1422
1422
1423
1423
1424
1424
1425
1425
1426
1426
1427
1427
1428
1428
1429
1429
1430
1430
1431
1431
1432
1432
1433
1433
1434
1434
1435
1435
1436
1436
1437
1437
1438
1438
1439
1439
1440
1440
1441
1441
1442
1442
1443
1443
1444
1444
1445
1445
1446
1446
1447
1447
1448
1448
1449
1449
1450
1450
1451
1451
1452
1452
1453
1453
1454
1454
1455
1455
1456
1456
1457
1457
1458
1458
1459
1459
1460
1460
1461
1461
1462
1462
1463
1463
1464
1464
1465
1465
1466
1466
1467
1467
1468
1468
1469
1469
1470
1470
1471
1471
1472
1472
1473
1473
1474
1474
1475
1475
1476
1476
1477
1477
1478
1478
1479
1479
1480
1480
1481
1481
1482
1482
1483
1483
1484
1484
1485
1485
1486
1486
1487
1487
1488
1488
1489
1489
1490
1490
1491
1491
1492
1492
1493
1493
1494
1494
1495
1495
1496
1496
1497
1497
1498
1498
1499
1499
1500
1500
1501
1501
1502
1502
1503
1503
1504
1504
1505
1505
1506
1506
1507
1507
1508
1508
1509
1509
1510
1510
1511
1511
1512
1512
1513
1513
1514
1514
1515
1515
1516
1516
1517
1517
1518
1518
1519
1519
1520
1520
1521
1521
1522
1522
1523
1523
1524
1524
1525
1525
1526
1526
1527
1527
1528
1528
152
```

```
353     private void ButtonDaftarRuanganActionPerformed(java.awt.event.ActionEvent evt) {  
354         // TODO add your handling code here:  
355         // 1. Tutup form saat ini  
356         this.dispose();  
357  
358         // 2. Buka form  
359         DaftarRuanganAdmin daftarruangan = new DaftarRuanganAdmin();  
360         daftarruangan.setVisible(true);  
361     }  
362  
363     private void ButtonPendaftaranPasienActionPerformed(java.awt.event.ActionEvent evt) {  
364         // TODO add your handling code here:  
365         // 1. Tutup form saat ini  
366         this.dispose();  
367  
368         // 2. Buka form  
369         PendaftaranPasienAdmin pendaftarpasien = new PendaftaranPasienAdmin();  
370         pendaftarpasien.setVisible(true);  
371     }  
372  
373     private void ButtonPendaftaranDokterActionPerformed(java.awt.event.ActionEvent evt) {  
374         // TODO add your handling code here:  
375         // 1. Tutup form saat ini  
376         this.dispose();  
377  
378         // 2. Buka form  
379         PendaftaranDokterAdmin daftardokter = new PendaftaranDokterAdmin();  
380         daftardokter.setVisible(true);  
381     }  
382  
383     private void ButtonPendaftaranObatActionPerformed(java.awt.event.ActionEvent evt) {  
384         // TODO add your handling code here:  
385         // 1. Tutup form saat ini  
386         this.dispose();  
387  
388         // 2. Buka form  
389         PendaftaranObatAdmin daftarobat = new PendaftaranObatAdmin();  
390         daftarobat.setVisible(true);  
391     }  
392  
393     private void ButtonJadwalDokterActionPerformed(java.awt.event.ActionEvent evt) {  
394         // TODO add your handling code here:  
395         // 1. Tutup form saat ini  
396         this.dispose();  
397  
398         // 2. Buka form  
399         JadwalDokterAdmin daftarjadwal = new JadwalDokterAdmin();  
400         daftarjadwal.setVisible(true);  
401     }  
402  
403     private void ButtonResepPasienActionPerformed(java.awt.event.ActionEvent evt) {  
404         // TODO add your handling code here:  
405         // 1. Tutup form saat ini  
406         this.dispose();  
407  
408         // 2. Buka form  
409         ResepPasienAdmin daftarresep = new ResepPasienAdmin();  
410         daftarresep.setVisible(true);  
411     }  
412  
413     private void ButtonLogoutActionPerformed(java.awt.event.ActionEvent evt) {  
414         // TODO add your handling code here:  
415         // Tampilkan dialog konfirmasi  
416         int pilihan = JOptionPane.showConfirmDialog(this,  
417             "Apakah Anda yakin ingin logout?",  
418             "Konfirmasi Logout",  
419             JOptionPane.YES_NO_OPTION);  
420  
421         // Jika pengguna memilih "Yes"  
422         if (pilihan == JOptionPane.YES_OPTION) {  
423             // 1. Tutup form saat ini  
424             this.dispose();  
425  
426             // 2. Buka form login  
427             Login login = new Login();  
428             login.setVisible(true);  
429         }  
430     }  
431 }
```

```
393     private void ButtonJadwalDokterActionPerformed(java.awt.event.ActionEvent evt) {  
394         // TODO add your handling code here:  
395         // 1. Tutup form saat ini  
396         this.dispose();  
397  
398         // 2. Buka form  
399         JadwalDokterAdmin daftarjadwal = new JadwalDokterAdmin();  
400         daftarjadwal.setVisible(true);  
401     }  
402  
403     private void ButtonResepPasienActionPerformed(java.awt.event.ActionEvent evt) {  
404         // TODO add your handling code here:  
405         // 1. Tutup form saat ini  
406         this.dispose();  
407  
408         // 2. Buka form  
409         ResepPasienAdmin daftarresep = new ResepPasienAdmin();  
410         daftarresep.setVisible(true);  
411     }  
412  
413     private void ButtonLogoutActionPerformed(java.awt.event.ActionEvent evt) {  
414         // TODO add your handling code here:  
415         // Tampilkan dialog konfirmasi  
416         int pilihan = JOptionPane.showConfirmDialog(this,  
417             "Apakah Anda yakin ingin logout?",  
418             "Konfirmasi Logout",  
419             JOptionPane.YES_NO_OPTION);  
420  
421         // Jika pengguna memilih "Yes"  
422         if (pilihan == JOptionPane.YES_OPTION) {  
423             // 1. Tutup form saat ini  
424             this.dispose();  
425  
426             // 2. Buka form login  
427             Login login = new Login();  
428             login.setVisible(true);  
429         }  
430     }  
431 }
```

```

432 private void ButtonDeletePasienActionPerformed(java.awt.event.ActionEvent evt) {
433 // TODO add your handling code here:
434 try {
435 Connection conn = koneksi.getKoneksi();
436 Statement stmt = conn.createStatement();
437 ResultSet rs = stmt.executeQuery("SELECT idPasien, nama FROM pasien");
438
439 JComboBox<String> comboBox = new JComboBox<>();
440 Map<String, String> pasienMap = new HashMap<>();
441
442 while (rs.next()) {
443     String id = rs.getString("idPasien");
444     String nama = rs.getString("nama");
445     comboBox.addItem(nama);
446     pasienMap.put(nama, id);
447 }
448
449 if (pasienMap.isEmpty()) {
450     JOptionPane.showMessageDialog(null, "Tidak ada pasien yang tersedia untuk dihapus.");
451     return;
452 }
453
454 int option = JOptionPane.showConfirmDialog(null, comboBox, "Pilih Pasien untuk Dihapus", JOptionPane.OK_CANCEL_OPTION);
455 if (option == JOptionPane.OK_OPTION) {
456     String namaDipilih = (String) comboBox.getSelectedItem();
457     String idDipilih = pasienMap.get(namaDipilih);
458
459     int konfirmasi = JOptionPane.showConfirmDialog(null, "Yakin ingin menghapus pasien '" + namaDipilih + "?", "Konfirmasi", JOptionPane.YES_NO_OPTION);
460     if (konfirmasi == JOptionPane.YES_OPTION) {
461         PasienDAO.deletePasien(idDipilih);
462         JOptionPane.showMessageDialog(null, "Pasien berhasil dihapus.");
463         tampilkanDataPasien(); // refresh tabel pasien
464     }
465
466     stmt.close();
467     conn.close();
468 } catch (Exception e) {
469     e.printStackTrace();
470     JOptionPane.showMessageDialog(null, "Terjadi kesalahan: " + e.getMessage());
471 }
472 }
473 }

475 private void ButtonEditPasienActionPerformed(java.awt.event.ActionEvent evt) {
476 // TODO add your handling code here:
477 try {
478     Connection conn = koneksi.getKoneksi();
479     Statement stmt = conn.createStatement();
480     ResultSet rs = stmt.executeQuery("SELECT idPasien, nama FROM pasien");
481
482     JComboBox<String> comboBox = new JComboBox<>();
483     Map<String, String> pasienMap = new HashMap<>();
484
485     while (rs.next()) {
486         String id = rs.getString("idPasien");
487         String nama = rs.getString("nama");
488         comboBox.addItem(nama);
489         pasienMap.put(nama, id);
490     }
491
492 if (pasienMap.isEmpty()) {
493     JOptionPane.showMessageDialog(null, "Tidak ada pasien yang tersedia untuk diedit.");
494     return;
495 }
496
497 int option = JOptionPane.showConfirmDialog(null, comboBox, "Pilih Pasien untuk Diedit", JOptionPane.OK_CANCEL_OPTION);
498 if (option == JOptionPane.OK_OPTION) {
499     String namaDipilih = (String) comboBox.getSelectedItem();
500     String idDipilih = pasienMap.get(namaDipilih);
501
502     // Ambil data lengkap pasien
503     PreparedStatement ps = conn.prepareStatement("SELECT * FROM pasien WHERE idPasien=?");
504     ps.setString(1, idDipilih);
505     ResultSet detail = ps.executeQuery();
506
507     if (detail.next()) {
508         String nama = detail.getString("nama");
509         String alamat = detail.getString("alamat");
510         String telepon = detail.getString("telepon");
511
512         JTextField namaField = new JTextField(nama);
513         JTextField alamatField = new JTextField(alamat);
514         JTextField teleponField = new JTextField(telepon);
515
516         Object[] fields = {
517             "Nama:", namaField,

```

```

516     Object[] fields = {
517         "Nama:", namaField,
518         "Alamat:", alamatField,
519         "Telepon:", teleponField
520     };
521
522     int edit = JOptionPane.showConfirmDialog(null, fields, "Edit Data Pasien", JOptionPane.OK_CANCEL_OPTION);
523     if (edit == JOptionPane.OK_OPTION) {
524         Pasien p = new Pasien(
525             idDipilih,
526             namaField.getText(),
527             alamatField.getText(),
528             teleponField.getText()
529         );
530
531         PasienDAO.updatePasien(p);
532         JOptionPane.showMessageDialog(null, "Data pasien berhasil diperbarui.");
533         tampilkanDataPasien(); // refresh tabel
534     }
535 }
536
537     detail.close();
538     ps.close();
539 }
540
541     stmt.close();
542     conn.close();
543 } catch (Exception e) {
544     e.printStackTrace();
545     JOptionPane.showMessageDialog(null, "Terjadi kesalahan: " + e.getMessage());
546 }
547 }

548 /**
549 * @param args the command line arguments
550 */
551 public static void main(String args[]) {
552     /* Set the Nimbus look and feel */
553     /* Look and feel setting code (optional) */
554
555     /* Create and display the form */
556     java.awt.EventQueue.invokeLater(new Runnable() {
557         public void run() {
558
559             /*
560             public static void main(String args[]) {
561                 /* Set the Nimbus look and feel */
562                 /* Look and feel setting code (optional) */
563
564                 /* Create and display the form */
565                 java.awt.EventQueue.invokeLater(new Runnable() {
566                     public void run() {
567                         new DaftarPasienAdmin().setVisible(true);
568                     }
569                 });
570             }
571
572             // Variables declaration - do not modify
573             private javax.swing.JButton ButtonDaftarDokter;
574             private javax.swing.JButton ButtonDaftarObat;
575             private javax.swing.JButton ButtonDaftarPasien;
576             private javax.swing.JButton ButtonDaftarRuang;
577             private javax.swing.JButton ButtonDashboard;
578             private javax.swing.JButton ButtonDeletePasien;
579             private javax.swing.JButton ButtonEditPasien;
580             private javax.swing.JButton ButtonJadwalDokter;
581             private javax.swing.JButton ButtonLogout;
582             private javax.swing.JButton ButtonPendaftaranDokter;
583             private javax.swing.JButton ButtonPendaftaranObat;
584             private javax.swing.JButton ButtonPendaftaranPasien;
585             private javax.swing.JButton ButtonResepPasien;
586             private javax.swing.JLabel LabelDaftarPasien;
587             private javax.swing.JPanel PanelSidebarAdmin2;
588             private javax.swing.JPanel PanelUtamaDaftar;
589             private javax.swing.JTable TabelDaftarPasien;
590             private javax.swing.JScrollPane jScrollPane;
591             // End of variables declaration
592         }
593     }
594 }
```

Penjelasan DaftarPasienAdmin.java

a. Deskripsi Umum

Kelas 'DaftarPasienAdmin.java' berfungsi sebagai halaman manajemen data pasien untuk admin. Halaman ini memungkinkan admin untuk melihat semua data pasien yang terdaftar, serta melakukan operasi pembaruan (edit) dan penghapusan (delete).

b. Fungsionalitas Utama

- Read: Menampilkan daftar semua pasien beserta ID, nama, alamat, dan telepon dalam sebuah 'JTable'.
- Update: Menyediakan tombol "Edit" untuk mengubah data pasien yang sudah ada.
- Delete: Menyediakan tombol "Delete" untuk menghapus data pasien dari database.

c. Komponen Penting (Code Breakdown)

- `tampilkanDataPasien()`: Metode ini bertugas mengambil daftar pasien dari database melalui 'PasienDAO.getAllPasien()'. Kemudian, data tersebut diiterasi dan setiap record pasien ditambahkan sebagai baris baru ke dalam 'DefaultTableModel' yang terhubung dengan 'TabelDaftarPasien'.
- `ButtonEditPasienActionPerformed(...)`: Logika untuk mengedit data pasien:
 - Mengambil daftar nama pasien dari database dan menampilkannya dalam 'JComboBox' di dalam sebuah 'JOptionPane'.
 - Setelah admin memilih pasien, data lengkap pasien tersebut diambil kembali dari database.
 - Sebuah dialog 'JOptionPane' baru muncul dengan beberapa 'JTextField' yang sudah terisi dengan data lama pasien (nama, alamat, telepon).
 - Jika admin mengonfirmasi perubahan, sebuah objek 'Pasien' baru dibuat dengan data yang telah diperbarui, dan 'PasienDAO.updatePasien()' dipanggil.
 - Tabel data pasien di-refresh dengan memanggil 'tampilkanDataPasien()'.
- `ButtonDeletePasienActionPerformed(...)`: Mekanisme penghapusan data:
 - Admin memilih pasien yang akan dihapus dari 'JComboBox'.
 - Sebuah dialog konfirmasi ditampilkan untuk memastikan tindakan penghapusan.
 - Jika dikonfirmasi, metode 'PasienDAO.deletePasien()' dipanggil dengan ID pasien yang dipilih.
 - Tabel akan di-refresh untuk menampilkan data terkini.

F. PendaftaranPasienAdmin.java

```
5  package klinik_pbo;
6
7  import controller.PasienController;
8  import dao.PasienDAO;
9  import java.awt.HeadlessException;
10 import java.sql.Connection;
11 import java.sql.PreparedStatement;
12 import java.sql.ResultSet;
13 import java.sql.SQLException;
14 import javax.swing.JOptionPane;
15 import model.Pasien;
16 import klinik_pbo.koneksai;
17
18 /**
19 * 
20 * @author avvjelly
21 */
22 public class PendaftaranPasienAdmin extends javax.swing.JFrame {
23
24 /**
25 * Creates new form PendaftaranPasienAdmin
26 */
27 public PendaftaranPasienAdmin() {
28     initComponents();
29 }
30
31 /**
32 * This method is called from within the constructor to initialize the form.
33 * WARNING: Do NOT modify this code. The content of this method is always
34 * regenerated by the Form Editor.
35 */
36 @SuppressWarnings("unchecked")
37 /**
38 * Generated Code
39 */
40
41 private void ButtonDashboardActionPerformed(java.awt.event.ActionEvent evt) {
42     // TODO add your handling code here:
43     // 1. Tutup form saat ini (DaftarPasienAdmin)
44     this.dispose();
45
46     // 2. Buka form DashboardAdmin
47     DashboardAdmin dashboard = new DashboardAdmin();
48     dashboard.setVisible(true);
49 }
50
51
52 private void ButtonDaftarPasienActionPerformed(java.awt.event.ActionEvent evt) {
53     // TODO add your handling code here:
54     // 1. Tutup form saat ini (DaftarPasienAdmin)
55     this.dispose();
56
57     // 2. Buka form DaftarPasien
58     DaftarPasienAdmin daftarpasien = new DaftarPasienAdmin();
59     daftarpasien.setVisible(true);
60 }
61
62 private void ButtonDaftarDokterActionPerformed(java.awt.event.ActionEvent evt) {
63     // TODO add your handling code here:
64     // 1. Tutup form saat ini (DaftarPasienAdmin)
65     this.dispose();
66
67     // 2. Buka form DaftarPasien
68     DaftarDokterAdmin daftardokter = new DaftarDokterAdmin();
69     daftardokter.setVisible(true);
70 }
71
72 private void ButtonDaftarObatActionPerformed(java.awt.event.ActionEvent evt) {
73     // TODO add your handling code here:
74     // 1. Tutup form saat ini
75     this.dispose();
76
77     // 2. Buka form
78     DaftarObatAdmin daftarobat = new DaftarObatAdmin();
79     daftarobat.setVisible(true);
80 }
81
82 private void TextfieldTeleponActionPerformed(java.awt.event.ActionEvent evt) {
83     // TODO add your handling code here:
84 }
85
86 private void ButtonDaftarRuanganActionPerformed(java.awt.event.ActionEvent evt) {
87     // TODO add your handling code here:
88     // 1. Tutup form saat ini
89     this.dispose();
90
91     // 2. Buka form
92     DaftarRuanganAdmin daftarruangan = new DaftarRuanganAdmin();
93     daftarruangan.setVisible(true);
94 }
```

```

388     private void ButtonSendPasienActionPerformed(java.awt.event.ActionEvent evt) {
389         // TODO add your handling code here:
390         try {
391             String id = TextfieldID.getText().trim();
392             String nama = TextfieldNama.getText().trim();
393             String alamat = TextareaAlamat.getText().trim();
394             String telepon = TextfieldTelepon.getText().trim();
395
396             // Validasi input
397             if (id.isEmpty() || nama.isEmpty() || alamat.isEmpty() || telepon.isEmpty()) {
398                 JOptionPane.showMessageDialog(this, "Semua field harus diisi terlebih dahulu.", "Peringatan", JOptionPane.WARNING_MESSAGE);
399                 return;
400             }
401
402             PasienController pc = new PasienController();
403             pc.tambahPasien(id, nama, alamat, telepon);
404
405             JOptionPane.showMessageDialog(this, "Data pasien berhasil disimpan!");
406
407             // Reset form
408             TextfieldID.setText("");
409             TextfieldNama.setText("");
410             TextareaAlamat.setText("");
411             TextfieldTelepon.setText("");
412
413         } catch (Exception ex) {
414             ex.printStackTrace();
415             JOptionPane.showMessageDialog(this, "Gagal menyimpan data pasien.\n" + ex.getMessage());
416         }
417
418     }
419
420     private void TextfieldIDActionPerformed (java.awt.event.ActionEvent evt) {
421         // TODO add your handling code here:
422     }
423
424     private void ButtonPendaftaranDokterActionPerformed (java.awt.event.ActionEvent evt) {
425         // TODO add your handling code here:
426         // 1. Tutup form saat ini
427         this.dispose();
428
429
430         // 2. Buka form
431         PendaftaranDokterAdmin daftardokter = new PendaftaranDokterAdmin();
432         daftardokter.setVisible(true);
433
434
435     private void ButtonPendaftaranObatActionPerformed (java.awt.event.ActionEvent evt) {
436         // TODO add your handling code here:
437         // 1. Tutup form saat ini
438         this.dispose();
439
440         // 2. Buka form
441         PendaftaranObatAdmin daftarobat = new PendaftaranObatAdmin();
442         daftarobat.setVisible(true);
443
444
445     private void ButtonJadwalDokterActionPerformed (java.awt.event.ActionEvent evt) {
446         // TODO add your handling code here:
447         // 1. Tutup form saat ini
448         this.dispose();
449
450         // 2. Buka form
451         JadwalDokterAdmin daftarjadwal = new JadwalDokterAdmin();
452         daftarjadwal.setVisible(true);
453
454
455     private void ButtonResepPasienActionPerformed (java.awt.event.ActionEvent evt) {
456         // TODO add your handling code here:
457         // 1. Tutup form saat ini
458         this.dispose();
459
460         // 2. Buka form
461         ResepPasienAdmin daftarresep = new ResepPasienAdmin();
462         daftarresep.setVisible(true);
463
464
465     private void ButtonLogoutActionPerformed (java.awt.event.ActionEvent evt) {
466         // TODO add your handling code here:
467         // Tampilkan dialog konfirmasi
468         int pilihan = JOptionPane.showConfirmDialog(this,
469             "Apakah Anda yakin ingin logout?",
470             "Konfirmasi Logout",
471             JOptionPane.YES_NO_OPTION);

```

```

474     if (pilihan == JOptionPane.YES_OPTION) {
475         // 1. Tutup form saat ini
476         this.dispose();
477
478         // 2. Buka form login
479         Login login = new Login();
480         login.setVisible(true);
481     }
482 }
483
484 /**
485 * @param args the command line arguments
486 */
487 public static void main(String args[]) {
488     /* Set the Nimbus look and feel */
489     /* Look and feel setting code (optional) */
490
491     /* Create and display the form */
492     java.awt.EventQueue.invokeLater(new Runnable() {
493         public void run() {
494             new PendaftaranPasienAdmin().setVisible(true);
495         }
496     });
497 }
498
499 // Variables declaration - do not modify
500 private javax.swing.JButton ButtonDaftarDokter;
501 private javax.swing.JButton ButtonDaftarObat;
502 private javax.swing.JButton ButtonDaftarPasien;
503 private javax.swing.JButton ButtonDaftarRuangan;
504 private javax.swing.JButton ButtonDashboard;
505 private javax.swing.JButton ButtonJadwalDokter;
506 private javax.swing.JButton ButtonLogout;
507 private javax.swing.JButton ButtonPendaftaranDokter;
508 private javax.swing.JButton ButtonPendaftaranObat;
509 private javax.swing.JButton ButtonPendaftaranPasien;
510 private javax.swing.JButton ButtonResepPasien;
511 private javax.swing.JButton ButtonSendPasien;
512 private javax.swing.JLabel LabelMasukkanAlamat;
513 private javax.swing.JLabel LabelMasukkanAlamat;
514 private javax.swing.JLabel LabelMasukkanID;
515 private javax.swing.JLabel LabelMasukkanNama;
516 private javax.swing.JLabel LabelPendaftaranPasien;
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545 }

```

Penjelasan PendaftaranPasienAdmin.java

a. Deskripsi Umum

`PendaftaranPasienAdmin.java` adalah kelas yang menyediakan antarmuka berupa formulir untuk admin guna mendaftarkan pasien baru ke dalam sistem klinik.

b. Fungsionalitas Utama

- Menyediakan field input untuk ID Pasien, Nama, Alamat ('`JTextArea`'), dan Nomor Telepon.
- Menyimpan data pasien yang baru diinput ke dalam tabel `pasien` di database.
- Memberikan validasi untuk memastikan semua field telah diisi sebelum data disimpan.

c. Komponen Penting (Code Breakdown)

- Konstruktor `public PendaftaranPasienAdmin()`: Hanya memanggil `initComponents()` untuk membangun antarmuka GUI yang telah dirancang.
- `ButtonSendPasienActionPerformed(...)`: Ini adalah *event handler* utama untuk tombol "Send".
 - Mengambil semua data dari `JTextField` dan `JTextArea`.

- Melakukan validasi sederhana untuk memeriksa apakah ada field yang kosong. Jika ya, sebuah `JOptionPane` peringatan akan muncul.
 - Membuat instance dari `PasienController` dan memanggil metode `tambahPasien` dengan melewatkannya data yang telah diinput. Penggunaan *Controller* di sini memisahkan logika bisnis penambahan data dari kode antarmuka.
 - Setelah berhasil, menampilkan pesan konfirmasi melalui `JOptionPane`.
 - Mengosongkan semua field input agar siap untuk pendaftaran berikutnya.
- Tombol Navigasi: Setiap tombol di sidebar berfungsi untuk menutup frame saat ini (`this.dispose()`) dan membuka frame lain yang sesuai, memungkinkan navigasi yang lancar di dalam aplikasi.

G. DaftarRuanganAdmin.java

```

5   package klinik_pbo;
6
7   import dao.RuangDAO;
8   import java.util.List;
9   import javax.swing.JOptionPane;
10  import javax.swing.table.DefaultTableModel;
11  import model.Ruangan;
12
13  /**
14   * 
15   * @author avvjelly
16   */
17  public class DaftarRuanganAdmin extends javax.swing.JFrame {
18      private void tampilkanDataRuangan () {
19          List<Ruangan> daftar = RuanganDAO.getAllRuangan();
20          DefaultTableModel model = (DefaultTableModel) TabelDaftarRuangan.getModel();
21          model.setRowCount(0); // Bersihkan isi tabel
22
23          for (Ruangan r : daftar) {
24              Object[] row = {
25                  r.getIdRuangan(),
26                  r.getNamaRuangan(),
27                  r.getLantai(),
28                  r.getKapasitas()
29              };
30              model.addRow(row);
31          }
32      }
33
34
35      /**
36       * Creates new form DaftarRuanganAdmin
37       */
38      public DaftarRuanganAdmin () {
39          initComponents ();
40          tampilkanDataRuangan ();
41      }

```

```
48     * @suppressWarnings("unchecked")
49     */
50     private void ButtonDashboardActionPerformed(java.awt.event.ActionEvent evt) {
51         // TODO add your handling code here:
52         // 1. Tutup form saat ini (DaftarPasienAdmin)
53         this.dispose();
54
55         // 3. Buka form DashboardAdmin
56         DashboardAdmin dashboard = new DashboardAdmin();
57         dashboard.setVisible(true);
58     }
59
60
61     private void ButtonDaftarPasienActionPerformed(java.awt.event.ActionEvent evt) {
62         // TODO add your handling code here:
63         // 1. Tutup form saat ini (DaftarPasienAdmin)
64         this.dispose();
65
66         // 2. Buka form DaftarPasien
67         DaftarPasienAdmin daftarpasien = new DaftarPasienAdmin();
68         daftarpasien.setVisible(true);
69     }
70
71     private void ButtonDaftarDokterActionPerformed(java.awt.event.ActionEvent evt) {
72         // TODO add your handling code here:
73         // 1. Tutup form saat ini (DaftarPasienAdmin)
74         this.dispose();
75
76         // 2. Buka form DaftarDokter
77         DaftarDokterAdmin daftardokter = new DaftarDokterAdmin();
78         daftardokter.setVisible(true);
79     }
80
81     private void ButtonDaftarObatActionPerformed(java.awt.event.ActionEvent evt) {
82         // TODO add your handling code here:
83         // 1. Tutup form saat ini
84         this.dispose();
85
86         // 2. Buka form
87         DaftarObatAdmin daftarobat = new DaftarObatAdmin();
88         daftarobat.setVisible(true);
89     }
90
91
92     private void ButtonPendaftaranPasienActionPerformed (java.awt.event.ActionEvent evt)
93     {
94         // TODO add your handling code here:
95         // 1. Tutup form saat ini
96         this.dispose();
97
98         // 2. Buka form
99         PendaftaranPasienAdmin pendaftarpasien = new PendaftaranPasienAdmin();
100        pendaftarpasien.setVisible(true);
101    }
102
103
104     private void ButtonPendaftaranDokterActionPerformed (java.awt.event.ActionEvent evt)
105    {
106        // TODO add your handling code here:
107        // 1. Tutup form saat ini
108        this.dispose();
109
110        // 2. Buka form
111        PendaftaranDokterAdmin daftardokter = new PendaftaranDokterAdmin();
112        daftardokter.setVisible(true);
113    }
114
115
116     private void ButtonPendaftaranObatActionPerformed (java.awt.event.ActionEvent evt) {
117        // TODO add your handling code here:
118        // 1. Tutup form saat ini
119        this.dispose();
120
121        // 2. Buka form
122        PendaftaranObatAdmin daftarobat = new PendaftaranObatAdmin();
123        daftarobat.setVisible(true);
124    }
125
126
127     private void ButtonJadwalDokterActionPerformed (java.awt.event.ActionEvent evt) {
128        // TODO add your handling code here:
129        // 1. Tutup form saat ini
130        this.dispose();
131
132        // 2. Buka form
133        JadwalDokterAdmin daftardjadwal = new JadwalDokterAdmin();
134        daftardjadwal.setVisible(true);
135    }
136
137
138
139
```

```

361     private void ButtonResepPasienActionPerformed(java.awt.event.ActionEvent evt) {
362         // TODO add your handling code here:
363         // 1. Tutup form saat ini
364         this.dispose();
365
366         // 2. Buka form
367         ResepPasienAdmin daftarresep = new ResepPasienAdmin();
368         daftarresep.setVisible(true);
369     }
370
371     private void ButtonLogoutActionPerformed(java.awt.event.ActionEvent evt) {
372         // TODO add your handling code here:
373         // Tampilkan dialog konfirmasi
374         int pilihan = JOptionPane.showConfirmDialog(this,
375             "Apakah Anda yakin ingin logout?",
376             "Konfirmasi Logout",
377             JOptionPane.YES_NO_OPTION);
378
379         // Jika pengguna memilih "Yes"
380         if (pilihan == JOptionPane.YES_OPTION) {
381             // 1. Tutup form saat ini
382             this.dispose();
383
384             // 2. Buka form login
385             Login login = new Login();
386             login.setVisible(true);
387         }
388     }
389
390     /**
391      * @param args the command line arguments
392     */
393     public static void main(String args[]) {
394         /* Set the Nimbus look and feel */
395         /* Look and feel setting code (optional) */
396
397         /* Create and display the form */
398         java.awt.EventQueue.invokeLater(new Runnable() {
399             public void run() {
400                 new DaftarRuanganAdmin().setVisible(true);
401             }
402         });
403     }
404
405     // Variables declaration - do not modify
406     private javax.swing.JButton ButtonDaftarDokter;
407     private javax.swing.JButton ButtonDaftarObat;
408     private javax.swing.JButton ButtonDaftarPasien;
409     private javax.swing.JButton ButtonDaftarRuang;
410     private javax.swing.JButton ButtonDashboard;
411     private javax.swing.JButton ButtonJadwalDokter;
412     private javax.swing.JButton ButtonLogout;
413     private javax.swing.JButton ButtonPendaftaranDokter;
414     private javax.swing.JButton ButtonPendaftaranObat;
415     private javax.swing.JButton ButtonPendaftaranPasien;
416     private javax.swing.JButton ButtonResepPasien;
417     private javax.swing.JLabel LabelDaftarRuang;
418     private javax.swing.JPanel PanelSidebarAdmin2;
419     private javax.swing.JPanel PanelUtamaDaftarRuang;
420     private javax.swing.JTable TabelDaftarRuang;
421     private javax.swing.JScrollPane jScrollPane;
422     // End of variables declaration
423 }
424

```

Penjelasan DaftarRuanganAdmin.java

a. Deskripsi Umum

Kelas DaftarRuanganAdmin.java menyediakan antarmuka bagi admin untuk melihat daftar ruangan yang tersedia di klinik. Berbeda dengan halaman "Daftar" lainnya, halaman ini bersifat read-only.

b. Fungsionalitas Utama

- Menampilkan daftar semua ruangan beserta ID, nama, lantai, dan kapasitasnya dalam sebuah JTable.
 - Tidak menyediakan fungsionalitas untuk menambah, mengedit, atau menghapus ruangan dari antarmuka ini. Tujuannya murni untuk penyajian informasi.

c. Komponen Penting (Code Breakdown)

- `tampilkanDataRuangan()`: Metode ini mengambil data semua ruangan dari database menggunakan `RuanganDAO.getAllRuangan()`. Data tersebut kemudian diisi ke dalam `TabelDaftarRuangan` baris per baris.
 - Antarmuka Pengguna: Desain GUI untuk frame ini secara sengaja tidak menyertakan tombol "Edit" atau "Delete", yang menegaskan perannya sebagai halaman informasional saja. Tombol navigasi sidebar tetap tersedia seperti biasa.

H. DashboardAdmin.java

```
1  /*
2   * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
3   * Click nbfs://nbhost/SystemFileSystem/Templates/GUIForms/JFrame.java to edit this template
4   */
5  package klinik_pbo;
6
7  import javax.swing.table.DefaultTableModel;
8  import dao.JadwalPraktikDAO;
9  import dao.UserDAO;
10 import model.JadwalPraktik;
11 import klinik_pbo.konaksi;
12 import java.sql.Connection;
13 import java.sql.DriverManager;
14 import java.sql.SQLException;
15 import java.util.List;
16 import java.util.logging.Level;
17 import java.util.logging.Logger;
18 import javax.swing.JOptionPane;
19 import model.User;
20
21 /**
22  *
23  * @author avvjelly
24  */
25 public class DashboardAdmin extends javax.swing.JFrame {
26     DashboardService service = new DashboardService(); // service buat ambil data
27
28
29
30 /**
31  * Creates new form DashboardAdmin
32  */
33 public DashboardAdmin() {
34     initComponents();
35     tampilkanJadwalPraktik();
36     tampilkanDaftarUser();
37
38     LabelAngkaPasien.setText(String.valueOf(service.getJumlahPasien()));
39     LabelAngkaDokter.setText(String.valueOf(service.getJumlahDokter()));
40     LabelAngkaRuangan.setText(String.valueOf(service.getJumlahRuangan()));
41 }
42
43 private void tampilkanDaftarUser() {
44     DefaultTableModel model = new DefaultTableModel(
45         new String[]{"ID User", "Username", "Password", "Role"},
46         0
47     );
48
49     List<User> users = UserDAO.getAllUsers();
50     for (User u : users) {
51         model.addRow(new Object[]{
52             u.getIdUser(),
53             u.getUsername(),
54             u.getPassword(),
55             u.getRole()
56         });
57     }
58     TabelDaftarUser.setModel(model);
59 }
```

```

62     private void tampilkanJadwalPraktik() {
63         DefaultTableModel model = new DefaultTableModel(new String[]{"ID Jadwal", "Hari", "Jam Mulai", "Jam Berakhir"}, 0);
64         try {
65             Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/klinik_db", "root", "");
66             JadwalPraktikDAO dao = new JadwalPraktikDAO(conn);
67             List<JadwalPraktik> list = dao.getAllJadwal();
68             for (JadwalPraktik jp : list) {
69                 model.addRow(new Object[]{
70                     jp.getIdJadwal(),
71                     jp.getHari(),
72                     jp.getJamMulai(),
73                     jp.getJamBerakhir()
74                 });
75             }
76             TabelJadwalPraktik.setModel(model);
77         } catch (SQLException e) {
78             JOptionPane.showMessageDialog(this, "Gagal mengambil data: " + e.getMessage());
79         }
80     }
81
82 }
83
84 /**
85 * This method is called from within the constructor to initialize the form.
86 * WARNING: Do NOT modify this code. The content of this method is always
87 * regenerated by the Form Editor.
88 */
89 @SuppressWarnings("unchecked")
90 Generated Code
487
488     private void ButtonDaftarPasienActionPerformed(java.awt.event.ActionEvent evt) {
489         // TODO add your handling code here:
490         // 1. Tutup form saat ini
491         this.dispose();
492
493         // 2. Buka form
494         DaftarPasienAdmin daftarpasien = new DaftarPasienAdmin();
495         daftarpasien.setVisible(true);
496     }
497
498     private void ButtonDaftarDokterActionPerformed(java.awt.event.ActionEvent evt) {
499         // TODO add your handling code here:
500         // 1. Tutup form saat ini
501         this.dispose();
502
503         // 2. Buka form
504         DaftarDokterAdmin daftardokter = new DaftarDokterAdmin();
505         daftardokter.setVisible(true);
506     }
507
508     private void ButtonDaftarObatActionPerformed(java.awt.event.ActionEvent evt) {
509         // TODO add your handling code here:
510         // 1. Tutup form saat ini
511         this.dispose();
512
513         // 2. Buka form
514         DaftarObatAdmin daftarobat = new DaftarObatAdmin();
515         daftarobat.setVisible(true);
516     }
517
518     private void ButtonDaftarRuanganActionPerformed(java.awt.event.ActionEvent evt) {
519         // TODO add your handling code here:
520         // 1. Tutup form saat ini
521         this.dispose();
522
523         // 2. Buka form
524         DaftarRuanganAdmin daftarruangan = new DaftarRuanganAdmin();
525         daftarruangan.setVisible(true);
526     }
527
528     private void ButtonPendaftaranPasienActionPerformed(java.awt.event.ActionEvent evt) {
529         // TODO add your handling code here:
530         // 1. Tutup form saat ini
531         this.dispose();
532
533         // 2. Buka form
534         PendaftaranPasienAdmin pendaftarpasien = new PendaftaranPasienAdmin();
535         pendaftarpasien.setVisible(true);
536     }

```

```
537  
538     private void ButtonPendaftaranDokterActionPerformed(java.awt.event.ActionEvent evt) {  
539         // TODO add your handling code here:  
540         // 1. Tutup form saat ini  
541         this.dispose();  
542  
543         // 2. Buka form  
544         PendaftaranDokterAdmin daftardokter = new PendaftaranDokterAdmin();  
545         daftardokter.setVisible(true);  
546     }  
547  
548     private void ButtonPendaftaranObatActionPerformed(java.awt.event.ActionEvent evt) {  
549         // TODO add your handling code here:  
550         // 1. Tutup form saat ini  
551         this.dispose();  
552  
553         // 2. Buka form  
554         PendaftaranObatAdmin daftarobat = new PendaftaranObatAdmin();  
555         daftarobat.setVisible(true);  
556     }  
557  
558     private void ButtonJadwalDokterActionPerformed(java.awt.event.ActionEvent evt) {  
559         // TODO add your handling code here:  
560         // 1. Tutup form saat ini  
561         this.dispose();  
562  
563         // 2. Buka form  
564         JadwalDokterAdmin daftarjadwal = new JadwalDokterAdmin();  
565         daftarjadwal.setVisible(true);  
566     }  
567  
568     private void ButtonResepPasienActionPerformed(java.awt.event.ActionEvent evt) {  
569         // TODO add your handling code here:  
570         // 1. Tutup form saat ini  
571         this.dispose();  
572  
573         // 2. Buka form  
574         ResepPasienAdmin daftarresep = new ResepPasienAdmin();  
575         daftarresep.setVisible(true);  
576     }
```

```
577  
578     private void ButtonLogoutActionPerformed(java.awt.event.ActionEvent evt) {  
579         // TODO add your handling code here:  
580         // Tampilkan dialog konfirmasi  
581         int pilihan = JOptionPane.showConfirmDialog(this,  
582             "Apakah Anda yakin ingin logout?",  
583             "Konfirmasi Logout",  
584             JOptionPane.YES_NO_OPTION);  
585  
586         // Jika pengguna memilih "Yes"  
587         if (pilihan == JOptionPane.YES_OPTION) {  
588             // 1. Tutup form saat ini  
589             this.dispose();  
590  
591             // 2. Buka form login  
592             Login login = new Login();  
593             login.setVisible(true);  
594         }  
595     }  
596  
597     /**  
598      * @param args the command line arguments  
599     */  
600     public static void main(String args[]) {  
601         /* Set the Nimbus look and feel */  
602         LookAndFeelSettingCode(optional);  
603  
604  
605         /* Create and display the form */  
606         java.awt.EventQueue.invokeLater(new Runnable() {  
607             public void run() {  
608                 new DashboardAdmin().setVisible(true);  
609             }  
610         });  
611     }  
612  
613     // Variables declaration - do not modify  
614     private javax.swing.JButton ButtonDaftarDokter;  
615     private javax.swing.JButton ButtonDaftarObat;
```

```

638     private javax.swing.JButton ButtonDaftarPasien;
639     private javax.swing.JButton ButtonDaftarRuangan;
640     private javax.swing.JButton ButtonDashboard;
641     private javax.swing.JButton ButtonJadwalDokter;
642     private javax.swing.JButton ButtonLogout;
643     private javax.swing.JButton ButtonPendaftaranDokter;
644     private javax.swing.JButton ButtonPendaftaranObat;
645     private javax.swing.JButton ButtonPendaftaranPasien;
646     private javax.swing.JButton ButtonResepPasien;
647     private javax.swing.JLabel LabelAngkaDokter;
648     private javax.swing.JLabel LabelAngkaPasien;
649     private javax.swing.JLabel LabelAngkaRuangan;
650     private javax.swing.JLabel LabelDaftarUser;
651     private javax.swing.JLabel LabelDokterDashboard;
652     private javax.swing.JLabel LabelJadwal;
653     private javax.swing.JLabel LabelPasienDashboard;
654     private javax.swing.JLabel LabelRuangDashboard;
655     private javax.swing.JLabel LabelTulisanDashboard;
656     private javax.swing.JPanel PanelAdminDashboard1;
657     private javax.swing.JPanel PanelAdminDashboard2;
658     private javax.swing.JPanel PanelAdminDashboard4;
659     private javax.swing.JPanel PanelSidebarAdmin;
660     private javax.swing.JPanel PanelUtama;
661     private javax.swing.JTable TabelDaftarUser;
662     private javax.swing.JTable TabelJadwalPraktik;
663     private javax.swing.JScrollPane jScrollPane;
664     private javax.swing.JScrollPane jScrollPane2;
665     // End of variables declaration
666 }
667

```

Penjelasan DasboardAdmin.java

a. Deskripsi Umum

DashboardAdmin.java adalah halaman utama (dashboard) untuk pengguna dengan peran "admin". Halaman ini menampilkan ringkasan data penting dari klinik dan berfungsi sebagai pusat navigasi ke semua fitur yang tersedia untuk admin.

b. Fungsionalitas Utama

- Menampilkan data statistik seperti jumlah total pasien, dokter, dan ruangan.
- Menampilkan daftar jadwal praktik dokter dalam sebuah tabel (TabelJadwalPraktik).
- Menampilkan daftar semua pengguna terdaftar (user) beserta perannya dalam tabel (TabelDaftarUser).
- Menyediakan tombol-tombol navigasi pada sidebar untuk mengakses halaman manajemen lainnya (Daftar Pasien, Dokter, Obat, Ruangan, Pendaftaran, Jadwal, Resep, dan Logout).

c. Komponen Penting (Code Breakdown)

- Konstruktor public DashboardAdmin(): Saat objek DashboardAdmin dibuat, konstruktor akan:
 - Memanggil initComponents() untuk membangun GUI.
 - Memanggil tampilkanJadwalPraktik() dan tampilkanDaftarUser() untuk mengisi tabel dengan data awal.
 - Menggunakan DashboardService untuk mengambil data agregat (jumlah pasien, dokter, ruangan) dan menampilkannya pada JLabel.

- `tampilkanDaftarUser()`: Mengambil daftar semua User dari `UserDAO.getAllUsers()`, lalu mengisi DefaultTableModel baris per baris dengan data ID, username, password, dan role, kemudian menampilkannya di TabelDaftarUser.
- `tampilkanJadwalPraktik()`: Mengambil data jadwal dari `JadwalPraktikDAO.getAllJadwal()`, lalu mengisi DefaultTableModel dan menampilkannya di TabelJadwalPraktik.
- Event Handlers untuk Tombol Navigasi: Setiap tombol pada sidebar (misal, `ButtonDaftarDokterActionPerformed`) memiliki logika sederhana: menutup frame saat ini (`this.dispose()`) dan membuka frame baru yang sesuai (misal, `new DaftarDokterAdmin().setVisible(true)`).
- `ButtonLogoutActionPerformed(...)`: Menampilkan dialog konfirmasi. Jika pengguna memilih "Yes", frame saat ini akan ditutup dan frame Login akan ditampilkan kembali.

I. DashboardDokter.java

```

5   package klinik_pbo;
6
7   import javax.swing.table.DefaultTableModel;
8   import dao.JadwalPraktikDAO;
9   import model.JadwalPraktik;
10  import klinik_pbo.koneksi;
11  import java.sql.Connection;
12  import java.sql.DriverManager;
13  import java.sql.SQLException;
14  import java.util.List;
15  import java.util.logging.Level;
16  import java.util.logging.Logger;
17  import javax.swing.JOptionPane;
18
19  /**
20  * @author avvjelly
21  */
22
23  public class DashboardDokter extends javax.swing.JFrame {
24
25  private void tampilkanJadwalPraktik() {
26      DefaultTableModel model = new DefaultTableModel(new String[]{"ID Jadwal", "Hari", "Jam Mulai", "Jam Berakhir"}, 0);
27      try {
28          Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/klinik_db", "root", "");
29          JadwalPraktikDAO dao = new JadwalPraktikDAO(conn);
30          List<JadwalPraktik> list = dao.getAllJadwal();
31          for (JadwalPraktik jp : list) {
32              model.addRow(new Object[]{
33                  jp.getIdJadwal(),
34                  jp.getHari(),
35                  jp.getJamMulai(),
36                  jp.getJamBerakhir()
37              });
38          }
39          TabelJadwalPraktik.setModel(model);
40      } catch (SQLException e) {
41          JOptionPane.showMessageDialog(this, "Gagal mengambil data: " + e.getMessage());
42      }
43  }

```

```
48     public DashboardDokter() {
49         initComponents();
50         tampilkanJadwalPraktik();
51     }
52
53
54     /**
55      * This method is called from within the constructor to initialize the form.
56      * WARNING: Do NOT modify this code. The content of this method is always
57      * regenerated by the Form Editor.
58     */
59     @SuppressWarnings("unchecked")
60     // Generated Code
61
62
63     private void ButtonDashboardActionPerformed(java.awt.event.ActionEvent evt) {
64         // TODO add your handling code here:
65         // 1. Tutup form saat ini
66         this.dispose();
67
68         // 2. Buka form
69         DashboardDokter dashboard = new DashboardDokter();
70         dashboard.setVisible(true);
71     }
72
73
74     private void ButtonJadwalDokterActionPerformed(java.awt.event.ActionEvent evt) {
75         // TODO add your handling code here:
76         // 1. Tutup form saat ini
77         this.dispose();
78
79         // 2. Buka form
80         JadwalDokterDokter daftarjadwal = new JadwalDokterDokter();
81         daftarjadwal.setVisible(true);
82     }
83
84
85     private void ButtonLogoutActionPerformed(java.awt.event.ActionEvent evt) {
86         // TODO add your handling code here:
87         // Tampilkan dialog konfirmasi
88         int pilihan = JOptionPane.showConfirmDialog(this,
89             "Apakah Anda yakin ingin logout?",
90             "Konfirmasi Logout",
91             JOptionPane.YES_NO_OPTION);
92
93         // Jika pengguna memilih "Yes"
94         if (pilihan == JOptionPane.YES_OPTION) {
95             // 1. Tutup form saat ini
96             this.dispose();
97
98             // 2. Buka form login
99             Login login = new Login();
100            login.setVisible(true);
101        }
102    }
103
104
105    private void ButtonResepPasienActionPerformed(java.awt.event.ActionEvent evt) {
106        // TODO add your handling code here:
107        // 1. Tutup form saat ini
108        this.dispose();
109
110        // 2. Buka form
111        ResepPasienDokter daftardesep = new ResepPasienDokter();
112        daftardesep.setVisible(true);
113    }
114
115
116    /**
117     * @param args the command line arguments
118     */
119
120    public static void main(String args[]) {
121        /* Set the Nimbus look and feel */
122        Look and feel setting code (optional)
123
124        /* Create and display the form */
125        java.awt.EventQueue.invokeLater(new Runnable() {
126            public void run() {
127                new DashboardDokter().setVisible(true);
128            }
129        });
130    }
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
```

```

297  ...
298 // Variables declaration - do not modify
299 private javax.swing.JButton ButtonDashboard;
300 private javax.swing.JButton ButtonJadwalDokter;
301 private javax.swing.JButton ButtonLogout;
302 private javax.swing.JButton ButtonResepPasien;
303 private javax.swing.JLabel LabelDashboardDokter;
304 private javax.swing.JLabel LabelJadwalPraktik;
305 private javax.swing.JPanel PanelSidebarAdmin2;
306 private javax.swing.JTable TabelJadwalPraktik;
307 private javax.swing.JPanel jPanel1;
308 private javax.swing.JScrollPane jScrollPane1;
309 // End of variables declaration
310 ...

```

Penjelasan DashboardDokter.java

a. Deskripsi Umum

DashboardDokter.java adalah halaman utama untuk pengguna dengan peran "dokter". Tampilannya lebih sederhana dibandingkan dashboard admin, hanya menampilkan informasi yang relevan bagi dokter.

b. Fungsionalitas Utama

- Menyambut dokter yang login.
- Menampilkan jadwal praktik klinik secara keseluruhan dalam sebuah tabel (TabelJadwalPraktik).
- Menyediakan navigasi ke fitur-fitur khusus dokter, yaitu melihat jadwal detail dan manajemen resep pasien.

c. Komponen Penting (Code Breakdown)\

- Konstruktor public DashboardDokter(): Memanggil initComponents() dan tampilkanJadwalPraktik() untuk mengisi tabel jadwal saat frame pertama kali dibuka.
- tampilkanJadwalPraktik(): Fungsinya identik dengan yang ada di DashboardAdmin, yaitu mengambil semua data jadwal praktik dan menampilkannya di JTable.
- Tombol Navigasi:
 - ButtonJadwalDokterActionPerformed: Mengarahkan ke frame JadwalDokterDokter.
 - ButtonResepPasienActionPerformed: Mengarahkan ke frame ResepPasienDokter.
 - ButtonLogoutActionPerformed: Mengembalikan pengguna ke halaman Login.

J. DashboardService.java

```
5     package klinik_pbo;
6
7     import java.sql.Connection;
8     import java.sql.DriverManager;
9     import java.sql.ResultSet;
10    import java.sql.Statement;
11
12    public class DashboardService {
13
14        public int getJumlahPasien() {
15            return getCount("pasien");
16        }
17
18        public int getJumlahDokter() {
19            return getCount("dokter");
20        }
21
22        public int getJumlahRuangan() {
23            return getCount("ruangan");
24        }
25
26        private int getCount(String tableName) {
27            int count = 0;
28            try (Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/klinik_db", "root", ""));
29            Statement stmt = conn.createStatement();
30            ResultSet rs = stmt.executeQuery("SELECT COUNT(*) AS total FROM " + tableName);
31            if (rs.next()) {
32                count = rs.getInt("total");
33            }
34        } catch (Exception e) {
35            System.out.println("Error counting from table " + tableName + ": " + e.getMessage());
36        }
37        return count;
38    }
39}
```

Penjelasan DashboardService.java

a. Deskripsi Umum

DashboardService.java adalah sebuah kelas layanan (service class) yang berfungsi untuk menyediakan data agregat atau ringkasan untuk ditampilkan di DashboardAdmin. Kelas ini mengisolasi logika query database untuk perhitungan data.

b. Fungsionalitas Utama

Menghitung jumlah total baris dari tabel tertentu di database.

Menyediakan metode publik untuk mendapatkan jumlah pasien, dokter, dan ruangan.

c. Komponen Penting (Code Breakdown)

- `getCount(String tableName)`: Ini adalah metode private helper yang menjadi inti dari kelas ini. Metode ini sangat efisien karena menggunakan kembali logika yang sama untuk menghitung record dari tabel manapun.
 - Menerima nama tabel sebagai parameter.
 - Menjalankan query SQL `SELECT COUNT(*) AS total FROM [tableName]`.
 - Menggunakan blok `try-with-resources` yang secara otomatis memastikan `Connection`, `Statement`, dan `ResultSet` ditutup setelah

digunakan, ini adalah praktik terbaik untuk mencegah kebocoran sumber daya (resource leak).

- Mengembalikan hasil hitungan sebagai int.
- Metode Publik (getJumlahPasien, getJumlahDokter, getJumlahRuang): Metode-metode ini hanya memanggil getCount dengan nama tabel yang sesuai, memberikan antarmuka yang bersih dan mudah dibaca bagi kelas lain (seperti DashboardAdmin) yang membutuhkannya.

K. JadwalDokterAdmin.java

```
5  package Klinik_pbo;
6
7  import controller.JadwalDokterController;
8  import dao.JadwalDokterDAO;
9  import java.util.List;
10 import javax.swing.JOptionPane;
11 import javax.swing.SwingUtilities;
12 import javax.swing.table.DefaultTableModel;
13 import model.JadwalDokter;
14
15 /**
16  * 
17  * @author avvjelly
18  */
19 public class JadwalDokterAdmin extends javax.swing.JFrame {
20     public void tampilkanDataJadwalDokter() {
21         // Pastikan dijalankan di EDT
22         SwingUtilities.invokeLater(() -> {
23             try {
24                 // Ambil data dari DAO
25                 List<JadwalDokter> daftar = JadwalDokterDAO.getAllJadwalDokterLengkap();
26
27                 // Bersihkan tabel
28                 DefaultTableModel model = (DefaultTableModel) TabelJadwalDokter.getModel();
29                 model.setRowCount(0);
30
31                 // Jika data kosong, tampilkan pesan
32                 if (daftar.isEmpty()) {
33                     JOptionPane.showMessageDialog(this, "Tidak ada data jadwal dokter.", "Info", JOptionPane.INFORMATION_MESSAGE);
34                     return;
35                 }
36
37                 // Isi tabel
38                 for (JadwalDokter jd : daftar) {
39                     model.addRow(new Object[]{
40                         jd.getIdJadwalDokter(),
41                         jd.getNamaDokter(),
42                         jd.getHari(),
43                         jd.getNamaRuangan()
44                     });
45                 }
46
47
48                 // Paksa update UI
49                 TabelJadwalDokter.revalidate();
50                 TabelJadwalDokter.repaint();
51
52             } catch (Exception e) {
53                 JOptionPane.showMessageDialog(this, "Gagal memuat data: " + e.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
54                 e.printStackTrace();
55             }
56         });
57     }
58
59 }
60
61 /**
62  * Creates new form JadwalPraktik
63  */
64 public JadwalDokterAdmin() {
65     initComponents();
66     tampilkanDataJadwalDokter();
67 }
68
69 /**
70  * This method is called from within the constructor to initialize the form.
71  * WARNING: Do NOT modify this code. The content of this method is always
72  * regenerated by the Form Editor.
73  */
74 @SuppressWarnings("unchecked")
75 Generated Code
76
77 private void ButtonDashboardActionPerformed(java.awt.event.ActionEvent evt) {
78     // TODO add your handling code here:
79     // 1. Tutup form saat ini (DaftarPasienAdmin)
80     this.dispose();
81
82     // 2. Buka form DashboardAdmin
83     DashboardAdmin dashboard = new DashboardAdmin();
84     dashboard.setVisible(true);
85 }
```

```
234     private void ButtonDaftarPasienActionPerformed(java.awt.event.ActionEvent evt) {  
235         // TODO add your handling code here:  
236         // 1. Tutup form saat ini (DaftarPasienAdmin)  
237         this.dispose();  
238  
239         // 2. Buka form DaftarPasien  
240         DaftarPasienAdmin daftarpasien = new DaftarPasienAdmin();  
241         daftarpasien.setVisible(true);  
242     }  
243  
244     private void ButtonDaftarDokterActionPerformed(java.awt.event.ActionEvent evt) {  
245         // TODO add your handling code here:  
246         // 1. Tutup form saat ini (DaftarPasienAdmin)  
247         this.dispose();  
248  
249         // 2. Buka form DaftarPasien  
250         DaftarDokterAdmin daftardokter = new DaftarDokterAdmin();  
251         daftardokter.setVisible(true);  
252     }  
253  
254     private void ButtonDaftarObatActionPerformed(java.awt.event.ActionEvent evt) {  
255         // TODO add your handling code here:  
256         // 1. Tutup form saat ini  
257         this.dispose();  
258  
259         // 2. Buka form  
260         DaftarObatAdmin daftarobat = new DaftarObatAdmin();  
261         daftarobat.setVisible(true);  
262     }  
263  
264     private void ButtonDaftarRuanganActionPerformed(java.awt.event.ActionEvent evt) {  
265         // TODO add your handling code here:  
266         // 1. Tutup form saat ini  
267         this.dispose();  
268  
269         // 2. Buka form  
270         DaftarRuanganAdmin daftarruangan = new DaftarRuanganAdmin();  
271         daftarruangan.setVisible(true);  
272     }  
273
```

```
274     private void ButtonResepPasienActionPerformed(java.awt.event.ActionEvent evt) {  
275         // TODO add your handling code here:  
276         // 1. Tutup form saat ini  
277         this.dispose();  
278  
279         // 2. Buka form  
280         ResepPasienAdmin daftardesep = new ResepPasienAdmin();  
281         daftardesep.setVisible(true);  
282     }  
283  
284     private void ButtonPendaftaranDokterActionPerformed(java.awt.event.ActionEvent evt) {  
285         // TODO add your handling code here:  
286         // 1. Tutup form saat ini  
287         this.dispose();  
288  
289         // 2. Buka form  
290         PendaftaranDokterAdmin daftardokter = new PendaftaranDokterAdmin();  
291         daftardokter.setVisible(true);  
292     }  
293  
294     private void ButtonPendaftaranPasienActionPerformed(java.awt.event.ActionEvent evt) {  
295         // TODO add your handling code here:  
296         // 1. Tutup form saat ini  
297         this.dispose();  
298  
299         // 2. Buka form  
300         PendaftaranPasienAdmin daftarpasien = new PendaftaranPasienAdmin();  
301         daftarpasien.setVisible(true);  
302     }  
303  
304     private void ButtonPendaftaranObatActionPerformed(java.awt.event.ActionEvent evt) {  
305         // TODO add your handling code here:  
306         // 1. Tutup form saat ini  
307         this.dispose();  
308  
309         // 2. Buka form  
310         PendaftaranObatAdmin daftarobat = new PendaftaranObatAdmin();  
311         daftarobat.setVisible(true);  
312     }  
313
```

```

414     private void ButtonLogoutActionPerformed(java.awt.event.ActionEvent evt) {
415         // TODO add your handling code here:
416         // Tampilkan dialog konfirmasi
417         int pilihan = JOptionPane.showConfirmDialog(this,
418             "Apakah Anda yakin ingin logout?",
419             "Konfirmasi Logout",
420             JOptionPane.YES_NO_OPTION);
421
422         // Jika pengguna memilih "Yes"
423         if (pilihan == JOptionPane.YES_OPTION) {
424             // 1. Tutup form saat ini
425             this.dispose();
426
427             // 2. Buka form login
428             Login login = new Login();
429             login.setVisible(true);
430         }
431     }
432
433     private void ButtonActionJadwalDokterActionPerformed(java.awt.event.ActionEvent evt) {
434         // TODO add your handling code here:
435         new JadwalDokterController().showActionDialog();
436         tampilanDataJadwalDokter(); // refresh tabel
437     }
438
439     /**
440      * @param args the command line arguments
441     */
442     public static void main(String args[]) {
443         /* Set the Nimbus look and feel */
444         Look and feel setting code (optional)
445
446         /* Create and display the form */
447         java.awt.EventQueue.invokeLater(new Runnable() {
448             public void run() {
449                 new JadwalDokterAdmin().setVisible(true);
450             }
451         });
452     }
453
454     // Variables declaration - do not modify
455     private javax.swing.JButton ButtonActionJadwalDokter;
456     private javax.swing.JButton ButtonDaftarDokter;
457     private javax.swing.JButton ButtonDaftarObat;
458     private javax.swing.JButton ButtonDaftarPasien;
459     private javax.swing.JButton ButtonDaftarRuangan;
460     private javax.swing.JButton ButtonDashboard;
461     private javax.swing.JButton ButtonJadwalDokter;
462     private javax.swing.JButton ButtonLogout;
463     private javax.swing.JButton ButtonPendaftaranDokter;
464     private javax.swing.JButton ButtonPendaftaranObat;
465     private javax.swing.JButton ButtonPendaftaranPasien;
466     private javax.swing.JButton ButtonResepPasien;
467     private javax.swing.JPanel PanelSidebarAdmin2;
468     private javax.swing.JTable TabelJadwalDokter;
469     private javax.swing.JLabel jLabel1;
470     private javax.swing.JPanel jPanel1;
471     private javax.swing.JScrollPane jScrollPane1;
472     // End of variables declaration
473 }

```

Penjelasan JadwalDokterAdmin.java

a. Deskripsi Umum

Halaman ini memungkinkan admin untuk mengelola penjadwalan dokter, yaitu membuat, melihat, memperbarui, dan menghapus jadwal praktik untuk setiap dokter di ruangan tertentu.

b. Fungsionalitas Utama

- Menampilkan daftar jadwal dokter yang sudah ada, lengkap dengan nama dokter, hari, dan ruangan.

- Menyediakan tombol "Action" yang memicu controller untuk menampilkan dialog CRUD (Create, Read, Update, Delete).

c. Komponen Penting (Code Breakdown)

- tampilkanDataJadwalDokter(): Mengambil daftar jadwal lengkap (hasil join dari beberapa tabel) melalui JadwalDokterDAO.getAllJadwalDokterLengkap() dan menampilkannya di TabelJadwalDokter.
- ButtonActionJadwalDokterActionPerformed(...): Berbeda dari form lain, tombol ini tidak langsung membuka dialog, melainkan memanggil new JadwalDokterController().showActionDialog(). Pola ini (menggunakan Controller) memisahkan logika bisnis dari tampilan. Controller tersebut akan menampilkan dialog JOptionPane berisi pilihan (Tambah, Edit, Hapus) dan menangani proses selanjutnya. Setelah aksi selesai, tampilkanDataJadwalDokter() dipanggil untuk me-refresh tabel.

L. JadwalDokterDokter.java

```

5   package klinik_pbo;
6
7   import dao.JadwalDokterDAO;
8   import javax.swing.table.DefaultTableModel;
9   import klinik_pbo.koneksi;
10  import java.sql.Connection;
11  import java.sql.DriverManager;
12  import java.sql.SQLException;
13  import java.util.List;
14  import java.util.logging.Level;
15  import java.util.logging.Logger;
16  import javax.swing.JOptionPane;
17  import javax.swing.SwingUtilities;
18  import model.JadwalDokter;
19
20 /**
21 *
22 * @author avvjelly
23 */
24 public class JadwalDokterDokter extends javax.swing.JFrame {
25     public void tampilkanDataJadwalDokter() {
26         // Pastikan dijalankan di EDT
27         SwingUtilities.invokeLater(() -> {
28             try {
29                 // Ambil data dari DAO
30                 List<JadwalDokter> daftar = JadwalDokterDAO.getAllJadwalDokterLengkap();
31
32                 // Bersihkan tabel
33                 DefaultTableModel model = (DefaultTableModel) TabelJadwalDokter.getModel();
34                 model.setRowCount(0);
35
36                 // Jika data kosong, tampilkan pesan
37                 if (daftar.isEmpty()) {
38                     JOptionPane.showMessageDialog(this, "Tidak ada data jadwal dokter.", "Info", JOptionPane.INFORMATION_MESSAGE);
39                     return;
40                 }
41
42                 // Isi tabel
43                 for (JadwalDokter jd : daftar) {
44                     model.addRow(new Object[]{
45                         jd.getIdJadwalDokter(),
46                         jd.getNamaDokter(),
47                         jd.getHari(),
48                     });
49                 }
50             } catch (Exception e) {
51                 e.printStackTrace();
52             }
53         });
54     }
55 }
```

```
48     | jd.getNamaRuangan()
49     | });
50   }
51
52   // Paksa update UI
53   TabelJadwalDokter.revalidate();
54   TabelJadwalDokter.repaint();
55
56 } catch (Exception e) {
57   JOptionPane.showMessageDialog(this, "Gagal memuat data: " + e.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
58   e.printStackTrace();
59 }
60 }
61
62 /**
63 * Creates new form JadwalDokterDokter
64 */
65 public JadwalDokterDokter() {
66   initComponents();
67   tampilanDataJadwalDokter();
68 }
69
70 /**
71 * This method is called from within the constructor to initialize the form.
72 * WARNING: Do NOT modify this code. The content of this method is always
73 * regenerated by the Form Editor.
74 */
75 @SuppressWarnings("unchecked")
76 Generated Code
77
78 private void ButtonDashboardActionPerformed(java.awt.event.ActionEvent evt) {
79
80   // TODO add your handling code here:
81   // 1. Tutup form saat ini
82   this.dispose();
83
84   // 2. Buka form
85   DashboardDokter dashboard = new DashboardDokter();
86   dashboard.setVisible(true);
87 }
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
598
599
599
600
601
602
603
604
605
606
607
608
609
609
610
611
612
613
614
615
616
617
618
619
619
620
621
622
623
624
625
626
627
627
628
629
629
630
631
632
633
634
635
636
637
638
639
639
640
641
642
643
644
645
646
647
648
649
649
650
651
652
653
654
655
656
657
658
659
659
660
661
662
663
664
665
666
667
668
669
669
670
671
672
673
674
675
676
677
678
679
679
680
681
682
683
684
685
686
687
687
688
689
689
690
691
692
693
694
695
696
697
697
698
699
699
700
701
702
703
704
705
706
707
708
709
709
710
711
712
713
714
715
716
717
717
718
719
719
720
721
722
723
724
725
726
727
727
728
729
729
730
731
732
733
734
735
736
737
737
738
739
739
740
741
742
743
744
745
746
747
747
748
749
749
750
751
752
753
754
755
756
757
757
758
759
759
760
761
762
763
764
765
766
767
767
768
769
769
770
771
772
773
774
775
776
777
777
778
779
779
780
781
782
783
784
785
786
787
787
788
789
789
790
791
792
793
794
795
796
797
797
798
799
799
800
801
802
803
804
805
806
807
808
809
809
810
811
812
813
814
815
816
817
817
818
819
819
820
821
822
823
824
825
826
827
827
828
829
829
830
831
832
833
834
835
836
837
837
838
839
839
840
841
842
843
844
845
846
847
847
848
849
849
850
851
852
853
854
855
856
857
857
858
859
859
860
861
862
863
864
865
866
867
867
868
869
869
870
871
872
873
874
875
876
877
877
878
879
879
880
881
882
883
884
885
886
887
887
888
889
889
890
891
892
893
894
895
896
897
897
898
899
899
900
901
902
903
904
905
906
907
907
908
909
909
910
911
912
913
914
915
916
916
917
918
918
919
920
921
922
923
924
925
926
926
927
928
928
929
930
931
932
933
934
935
936
937
937
938
939
939
940
941
942
943
944
945
946
946
947
948
948
949
950
951
952
953
954
955
956
957
957
958
959
959
960
961
962
963
964
965
966
966
967
968
968
969
970
971
972
973
974
975
976
976
977
978
978
979
980
981
982
983
984
985
986
987
987
988
989
989
990
991
992
993
994
995
996
997
997
998
999
999
1000
1000
1001
1002
1003
1004
1005
1006
1007
1007
1008
1009
1009
1010
1011
1012
1013
1014
1015
1015
1016
1017
1017
1018
1019
1019
1020
1021
1022
1023
1024
1025
1026
1026
1027
1028
1028
1029
1030
1031
1032
1033
1034
1035
1036
1036
1037
1038
1038
1039
1040
1041
1042
1043
1044
1045
1045
1046
1047
1047
1048
1049
1049
1050
1051
1052
1053
1054
1055
1056
1056
1057
1058
1058
1059
1060
1061
1062
1063
1064
1065
1065
1066
1067
1067
1068
1069
1069
1070
1071
1072
1073
1074
1075
1075
1076
1077
1077
1078
1079
1079
1080
1081
1082
1083
1084
1085
1085
1086
1087
1087
1088
1089
1089
1090
1091
1092
1093
1093
1094
1095
1095
1096
1097
1097
1098
1099
1099
1100
1101
1102
1103
1103
1104
1105
1105
1106
1107
1107
1108
1109
1109
1110
1111
1112
1113
1113
1114
1115
1115
1116
1117
1117
1118
1119
1119
1120
1121
1122
1123
1123
1124
1125
1125
1126
1127
1127
1128
1129
1129
1130
1131
1132
1133
1133
1134
1135
1135
1136
1137
1137
1138
1139
1139
1140
1141
1142
1143
1143
1144
1145
1145
1146
1147
1147
1148
1149
1149
1150
1151
1152
1153
1153
1154
1155
1155
1156
1157
1157
1158
1159
1159
1160
1161
1162
1163
1163
1164
1165
1165
1166
1167
1167
1168
1169
1169
1170
1171
1172
1173
1173
1174
1175
1175
1176
1177
1177
1178
1179
1179
1180
1181
1182
1183
1183
1184
1185
1185
1186
1187
1187
1188
1189
1189
1190
1191
1192
1193
1193
1194
1195
1195
1196
1197
1197
1198
1199
1199
1200
1201
1202
1203
1203
1204
1205
1205
1206
1207
1207
1208
1209
1209
1210
1211
1212
1213
1213
1214
1215
1215
1216
1217
1217
1218
1219
1219
1220
1221
1222
1223
1223
1224
1225
1225
1226
1227
1227
1228
1229
1229
1230
1231
1232
1233
1233
1234
1235
1235
1236
1237
1237
1238
1239
1239
1240
1241
1242
1243
1243
1244
1245
1245
1246
1247
1247
1248
1249
1249
1250
1251
1252
1253
1253
1254
1255
1255
1256
1257
1257
1258
1259
1259
1260
1261
1262
1263
1263
1264
1265
1265
1266
1267
1267
1268
1269
1269
1270
1271
1272
1273
1273
1274
1275
1275
1276
1277
1277
1278
1279
1279
1280
1281
1282
1283
1283
1284
1285
1285
1286
1287
1287
1288
1289
1289
1290
1291
1292
1293
1293
1294
1295
1295
1296
1297
1297
1298
1299
1299
1300
1301
1302
1303
1303
1304
1305
1305
1306
1307
1307
1308
1309
1309
1310
1311
1312
1313
1313
1314
1315
1315
1316
1317
1317
1318
1319
1319
1320
1321
1322
1323
1323
1324
1325
1325
1326
1327
1327
1328
1329
1329
1330
1331
1332
1333
1333
1334
1335
1335
1336
1337
1337
1338
1339
1339
1340
1341
1342
1343
1343
1344
1345
1345
1346
1347
1347
1348
1349
1349
1350
1351
1352
1353
1353
1354
1355
1355
1356
1357
1357
1358
1359
1359
1360
1361
1362
1363
1363
1364
1365
1365
1366
1367
1367
1368
1369
1369
1370
1371
1372
1373
1373
1374
1375
1375
1376
1377
1377
1378
1379
1379
1380
1381
1382
1383
1383
1384
1385
1385
1386
1387
1387
1388
1389
1389
1390
1391
1392
1393
1393
1394
1395
1395
1396
1397
1397
1398
1399
1399
1400
1401
1402
1403
1403
1404
1405
1405
1406
1407
1407
1408
1409
1409
1410
1411
1412
1413
1413
1414
1415
1415
1416
1417
1417
1418
1419
1419
1420
1421
1422
1423
1423
1424
1425
1425
1426
1427
1427
1428
1429
1429
1430
1431
1432
1433
1433
1434
1435
1435
1436
1437
1437
1438
1439
1439
1440
1441
1442
1443
1443
1444
1445
1445
1446
1447
1447
1448
1449
1449
1450
1451
1452
1453
1453
1454
1455
1455
1456
1457
1457
1458
1459
1459
1460
1461
1462
1463
1463
1464
1465
1465
1466
1467
1467
1468
1469
1469
1470
1471
1472
1473
1473
1474
1475
1475
1476
1477
1477
1478
1479
1479
1480
1481
1482
1483
1483
1484
1485
1485
1486
1487
1487
1488
1489
1489
1490
1491
1492
1493
1493
1494
1495
1495
1496
1497
1497
1498
1499
1499
1500
1501
1502
1503
1503
1504
1505
1505
1506
1507
1507
1508
1509
1509
1510
1511
1512
1513
1513
1514
1515
1515
1516
1517
1517
1518
1519
1519
1520
1521
1522
1523
1523
1524
1525
1525
1526
1527
1527
1528
1529
1529
1530
1531
1532
1533
1533
1534
1535
1535
1536
1537
1537
1538
1539
1539
1540
1541
1542
1543
1543
1544
1545
1545
1546
1547
1547
1548
1549
1549
1550
1551
1552
1553
1553
1554
1555
1555
1556
1557
1557
1558
1559
1559
1560
1561
1562
1563
1563
1564
1565
1565
1566
1567
1567
1568
1569
1569
1570
1571
1572
1573
1573
1574
1575
1575
1576
1577
1577
1578
1579
1579
1580
1581
1582
1583
1583
1584
1585
1585
1586
1587
1587
1588
1589
1589
1590
1591
1592
1593
1593
1594
1595
1595
1596
1597
1597
1598
1599
1599
1600
1601
1602
1603
1603
1604
1605
1605
1606
1607
1607
1608
1609
1609
1610
1611
1612
1613
1613
1614
1615
1615
1616
1617
1617
1618
1619
1619
1620
1621
1622
1623
1623
1624
1625
1625
1626
1627
1627
1628
1629
1629
1630
1631
1632
1633
1633
1634
1635
1635
1636
1637
1637
1638
1639
1639
1640
1641
1642
1643
1643
1644
1645
1645
1646
1647
1647
1648
1649
1649
1650
1651
1652
1653
1653
1654
1655
1655
1656
1657
1657
1658
1659
1659
1660
1661
1662
1663
1663
1664
1665
1665
1666
1667
1667
1668
1669
1669
1670
1671
1672
1673
1673
1674
1675
1675
1676
1677
1677
1678
1679
1679
1680
1681
1682
1683
1683
1684
1685
1685
1686
1687
1687
1688
1689
1689
1690
1691
1692
1693
1693
1694
1695
1695
1696
1697
1697
1698
1699
1699
1700
1701
1702
1703
1703
1704
1705
1705
1706
1707
1707
1708
1709
1709
1710
1711
1712
1713
1713
1714
1715
1715
1716
1717
1717
1718
1719
1719
1720
1721
1722
1723
1723
1724
1725
1725
1726
1727
1727
1728
1729
1729
1730
1731
1732
1733
1733
1734
1735
1735
1736
1737
1737
1738
1739
1739
1740
1741
1742
1743
1743
1744
1745
1745
1746
1747
1747
1748
1749
1749
1750
1751
1752
1753
1753
1754
1755
1755
1756
1757
1757
1758
1759
1759
1760
1761
1762
1763
1763
1764
1765
1765
1766
1767
1767
1768
1769
1769
1770
1771
1772
1773
1773
1774
1775
1775
1776
1777
1777
1778
1779
1779
1780
1781
1782
1783
1783
1784
1785
1785
1786
1787
1787
1788
1789
1789
1790
1791
1792
1793
1793
1794
1795
1795
1796
1797
1797
1798
1799
1799
1800
1801
1802
1803
1803
1804
1805
1805
1806
1807
1807
1808
1809
1809
1810
1811
1812
1813
1813
1814
1815
1815
1816
1817
1817
1818
1819
1819
1820
1821
1822
1823
1823
1824
1825
1825
1826
1827
1827
1828
1829
1829
1830
1831
1832
1833
1833
1834
1835
1835
1836
1837
1837
1838
1839
1839
1840
1841
1842
1843
1843
1844
1845
1845
1846
1847
1847
1848
1849
1849
1850
1851
1852
1853
1853
1854
1855
1855
1856
1857
1857
1858
1859
1859
1860
1861
1862
1863
1863
1864
1865
1865
1866
1867
1867
1868
1869
1869
1870
1871
1872
1873
1873
1874
1875
1875
1876
1877
1877
1878
1879
1879
1880
1881
1882
1883
1883
1884
1885
1885
1886
1887
1887
1888
1889
1889
1890
1891
1892
1893
1893
1894
1895
1895
1896
1897
1897
1898
1899
1899
1900
1901
1902
1903
1903
1904
1905
1905
1906
1907
1907
1908
1909
1909
1910
1911
1912
1913
1913
1914
1915
1915
1916
1917
1917
1918
1919
1919
1920
1921
1922
1923
1923
1924
1925
1925
1926
1927
1927
1928
1929
1929
1930
1931
1932
1933
1933
1934
1935
1935
1936
1937
1937
1938
1939
1939
1940
1941
1942
1943
1943
1944
1945
1945
1946
1947
1947
1948
1949
1949
1950
1951
1952
1953
1953
1954
1955
1955
1956
1957
1957
1958
1959
1959
1960
1961
1962
1963
1963
1964
1965
1965
1966
1967
1967
1968
1969
1969
1970
1971
1972
1973
1973
1974
1975
1975
1976
1977
1977
1978
1979
1979
1980
1981
1982
1983
1983
1984
1985
1985
1986
1987
1987
1988
1989
1989
1990
1991
1992
1993
1993
1994
1995
1
```

Penjelasan JadwalDokterDokter.java

a. Deskripsi Umum

Halaman ini ditujukan untuk dokter agar dapat melihat jadwal praktik mereka sendiri atau jadwal dokter lain.

b. Fungsionalitas Utama

- Hanya untuk melihat (read-only) daftar jadwal praktik dokter.
- Menampilkan ID Jadwal, Nama Dokter, detail Jadwal (hari, jam), dan Ruangan.

c. Komponen Penting (Code Breakdown)

tampilkanDataJadwalDokter(): Fungsinya sama seperti di JadwalDokterAdmin, yaitu mengambil dan menampilkan data jadwal lengkap ke dalam JTable. Tidak ada tombol aksi seperti edit atau delete, menegaskan sifatnya yang read-only bagi dokter.

M. Login.java

```
package klinik_pbo;

import java.awt.HeadlessException;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import javax.swing.JOptionPane;

/*
 * @author avvjelly
 */
public class Login extends javax.swing.JFrame {

    /*
     * Creates new form Login
     */
    public Login() {
        initComponents();
    }

    /*
     * This method is called from within the constructor to initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // Generated Code
    private void ButtonLoginActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        String username = FieldUsername.getText().trim();
        String password = FieldPassword.getText().trim();

        if (username.isEmpty() || password.isEmpty()) {
            JOptionPane.showMessageDialog(this, "Username dan Password tidak boleh kosong!", "Peringatan", JOptionPane.WARNING_MESSAGE);
            return;
        }
    }
}
```

```

try {
    Connection conn = koneksi.getKoneksi(); // Pastikan class koneksi.java ada
    String sql = "SELECT * FROM user WHERE LOWER(username) = ? AND password = ?";

    PreparedStatement pst = conn.prepareStatement(sql);
    pst.setString(1, username);
    pst.setString(2, password);

    ResultSet rs = pst.executeQuery();

    if (rs.next()) {
        String role = rs.getString("role");
        JOptionPane.showMessageDialog(this, "Login berhasil sebagai " + role, "Sukses", JOptionPane.INFORMATION_MESSAGE);

        // Redirect sesuai role
        if ("admin".equals(role)) {
            DashboardAdmin admin = new DashboardAdmin();
            admin.setVisible(true);
        } else if ("dokter".equals(role)) {
            DashboardDokter dokter = new DashboardDokter();
            dokter.setVisible(true);
        } else {
            JOptionPane.showMessageDialog(this, "Role tidak dikenali!", "Error", JOptionPane.ERROR_MESSAGE);
        }

        this.dispose(); // Tutup form login
    } else {
        JOptionPane.showMessageDialog(this, "Username atau Password salah!", "Gagal", JOptionPane.ERROR_MESSAGE);
    }
}

} catch (HeadlessException | SQLException e) {
    JOptionPane.showMessageDialog(this, "Terjadi kesalahan: " + e.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
}
}

/**
 * @param args the command line arguments
 */
public static void main(String args[]) {
    /* Set the Nimbus look and feel */
    [Look and feel setting code (optional)]

    /* Create and display the form */
    [
    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        [Look and feel setting code (optional)]

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new Login().setVisible(true);
            }
        });
    }
}
]

    // Variables declaration - do not modify
    private javax.swing.JButton ButtonLogin;
    private javax.swing.JPasswordField FieldPassword;
    private javax.swing.JTextField FieldUsername;
    private javax.swing.JPanel FrameLoginKanan;
    private javax.swing.JPanel FrameLoginKiri;
    private javax.swing.JLabel LabelForm;
    private javax.swing.JLabel LabelPassword;
    private javax.swing.JLabel LabelUsername;
    private javax.swing.JLabel Logo;
    // End of variables declaration
}

```

Penjelasan Login.java

a. Deskripsi Umum

Login.java adalah kelas yang bertanggung jawab untuk antarmuka login pengguna. Kelas ini menjadi gerbang utama aplikasi, di mana pengguna (admin atau dokter) harus memasukkan username dan password untuk dapat mengakses sistem.

b. Fungsionalitas Utama

- Menyediakan field untuk input username dan password.

- Memvalidasi kredensial yang dimasukkan oleh pengguna terhadap data yang ada di tabel user pada database.
- Memberikan notifikasi jika login berhasil atau gagal.
- Mengarahkan pengguna ke dashboard yang sesuai dengan perannya (role) setelah login berhasil, yaitu DashboardAdmin untuk admin dan DashboardDokter untuk dokter.

c. Komponen Penting (Code Breakdown)

- `initComponents()`: Metode ini (dihasilkan oleh NetBeans GUI Builder) menginisialisasi semua komponen GUI seperti JTextField untuk username, JPasswordField untuk password, dan JButton untuk tombol login.
- `ButtonLoginActionPerformed(java.awt.event.ActionEvent evt)`: Ini adalah event handler untuk tombol login. Logikanya adalah sebagai berikut:
 - Mengambil teks dari FieldUsername dan FieldPassword.
 - Melakukan validasi dasar untuk memastikan kedua field tidak kosong.
 - Membangun koneksi ke database menggunakan `koneksi.getKoneksi()`.
 - Mempersiapkan dan mengeksekusi query SQL (`SELECT * FROM user WHERE LOWER(username) = ? AND password = ?`) menggunakan `PreparedStatement` untuk keamanan (mencegah SQL Injection).
 - Jika query mengembalikan hasil (`rs.next()` bernilai true), artinya username dan password cocok.
 - Membaca kolom role dari hasil query.
- Berdasarkan role:
 - Jika "admin", maka form Login ditutup (`this.dispose()`) dan DashboardAdmin dibuka.
 - Jika "dokter", maka form Login ditutup dan DashboardDokter dibuka.
 - Jika query tidak mengembalikan hasil, sebuah JOptionPane akan ditampilkan untuk memberitahu bahwa username atau password salah.

N. ResepPasienAdmin.java

```
5 package klinik_pbo;
6
7 import controller.ResepPasienController;
8 import dao.ResepPasienDAO;
9 import java.util.List;
10 import javax.swing.JOptionPane;
11 import javax.swing.table.DefaultTableModel;
12 import model.ResepPasien;
13
14 /**
15 * @author avvjelly
16 */
17
18 public class ResepPasienAdmin extends javax.swing.JFrame {
19     private void tampilkanDataResepPasien() {
20         List<ResepPasien> daftar = ResepPasienDAO.getAllResepLengkap();
21         DefaultTableModel model = (DefaultTableModel) TabelResepPasien.getModel();
22         model.setRowCount(0); // Bersihkan isi tabel
23
24         for (ResepPasien r : daftar) {
25             Object[] row = {
26                 r.getIdResep(),
27                 r.getPenyakit(),
28                 r.getTanggal(),
29                 r.getNamaPasien(),
30                 r.getNamaDokter(),
31                 r.getNamaObat(),
32                 r.getStatus()
33             };
34             model.addRow(row);
35         }
36     }
37
38 }
39
40 /**
41 * Creates new form ResepPasienAdmin
42 */
43 public ResepPasienAdmin() {
44     initComponents();
45     tampilkanDataResepPasien();
46 }
```

```
53 @SuppressWarnings("unchecked")
54 Generated Code
55
56 private void ButtonDashboardActionPerformed(java.awt.event.ActionEvent evt) {
57     // TODO add your handling code here:
58     // 1. Tutup form saat ini (DaftarPasienAdmin)
59     this.dispose();
60
61     // 2. Buka form DashboardAdmin
62     DashboardAdmin dashboard = new DashboardAdmin();
63     dashboard.setVisible(true);
64 }
65
66 private void ButtonDaftarPasienActionPerformed(java.awt.event.ActionEvent evt) {
67     // TODO add your handling code here:
68     // 1. Tutup form saat ini (DaftarPasienAdmin)
69     this.dispose();
70
71     // 2. Buka form DaftarPasien
72     DaftarPasienAdmin daftarpasien = new DaftarPasienAdmin();
73     daftarpasien.setVisible(true);
74 }
75
76 private void ButtonDaftarDokterActionPerformed(java.awt.event.ActionEvent evt) {
77     // TODO add your handling code here:
78     // 1. Tutup form saat ini (DaftarPasienAdmin)
79     this.dispose();
80
81     // 2. Buka form DaftarPasien
82     DaftarDokterAdmin daftardokter = new DaftarDokterAdmin();
83     daftardokter.setVisible(true);
84 }
85
86 private void ButtonDaftarObatActionPerformed(java.awt.event.ActionEvent evt) {
87     // TODO add your handling code here:
88     // 1. Tutup form saat ini
89     this.dispose();
90
91     // 2. Buka form
92     DaftarObatAdmin daftarobat = new DaftarObatAdmin();
93     daftarobat.setVisible(true);
94 }
```

```
343     private void ButtonDaftarRuanganActionPerformed(java.awt.event.ActionEvent evt) {  
344         // TODO add your handling code here:  
345         // 1. Tutup form saat ini  
346         this.dispose();  
347  
348         // 2. Buka form  
349         DaftarRuanganAdmin daftarruangan = new DaftarRuanganAdmin();  
350         daftarruangan.setVisible(true);  
351     }  
352  
353     private void ButtonResepPasienActionPerformed (java.awt.event.ActionEvent evt) {  
354         // TODO add your handling code here:  
355         // 1. Tutup form saat ini  
356         this.dispose();  
357  
358         // 2. Buka form  
359         PendaftaranDokterAdmin daftardokter = new PendaftaranDokterAdmin();  
360         daftardokter.setVisible(true);  
361     }  
362  
363     private void ButtonPendaftaranPasienActionPerformed (java.awt.event.ActionEvent evt) {  
364         // TODO add your handling code here:  
365         // 1. Tutup form saat ini  
366         this.dispose();  
367  
368         // 2. Buka form  
369         PendaftaranPasienAdmin daftarpasien = new PendaftaranPasienAdmin();  
370         daftarpasien.setVisible(true);  
371     }  
372  
373     private void ButtonPendaftaranObatActionPerformed(java.awt.event.ActionEvent evt) {  
374         // TODO add your handling code here:  
375         // 1. Tutup form saat ini  
376         this.dispose();  
377  
378         // 2. Buka form  
379         PendaftaranObatAdmin daftarobat = new PendaftaranObatAdmin();  
380         daftarobat.setVisible(true);  
381     }  
382
```

```
283     private void ButtonJadwalDokterActionPerformed(java.awt.event.ActionEvent evt) {  
284         // TODO add your handling code here:  
285         // 1. Tutup form saat ini  
286         this.dispose();  
287  
288         // 2. Buka form  
289         JadwalDokterAdmin daftardjadwal = new JadwalDokterAdmin();  
290         daftardjadwal.setVisible(true);  
291     }  
292  
293     private void ButtonLogoutActionPerformed(java.awt.event.ActionEvent evt) {  
294         // TODO add your handling code here:  
295         // Tampilkan dialog konfirmasi  
296         int pilihan = JOptionPane.showConfirmDialog(this,  
297             "Apakah Anda yakin ingin logout?",  
298             "Konfirmasi Logout",  
299             JOptionPane.YES_NO_OPTION);  
300  
301         // Jika pengguna memilih "Yes"  
302         if (pilihan == JOptionPane.YES_OPTION) {  
303             // 1. Tutup form saat ini  
304             this.dispose();  
305  
306             // 2. Buka form login  
307             Login login = new Login();  
308             login.setVisible(true);  
309         }  
310     }  
311  
312     private void ButtonActionResepPasienActionPerformed(java.awt.event.ActionEvent evt) {  
313         // TODO add your handling code here:  
314         new ResepPasienController("admin").showActionDialog();  
315         tampilanDataResepPasien(); // refresh tabel jika kamu punya  
316  
317     }  
318  
319     /**  
320      * @param args the command line arguments  
321     */  
322     public static void main(String args[]) {  
323         /* Set the Nimbus look and feel */  
324         /* Look and feel setting code (optional) */  
325     }  
326 }
```



```
446     /* Create and display the form */
447     java.awt.EventQueue.invokeLater(new Runnable() {
448         public void run() {
449             new ResepPasienAdmin().setVisible(true);
450         }
451     });
452 }
453
454 // Variables declaration - do not modify
455 private javax.swing.JButton ButtonActionResepPasien;
456 private javax.swing.JButton ButtonDaftarDokter;
457 private javax.swing.JButton ButtonDaftarObat;
458 private javax.swing.JButton ButtonDaftarPasien;
459 private javax.swing.JButton ButtonDaftarRuangan;
460 private javax.swing.JButton ButtonDashboard;
461 private javax.swing.JButton ButtonJadwalDokter;
462 private javax.swing.JButton ButtonLogout;
463 private javax.swing.JButton ButtonPendaftaranDokter;
464 private javax.swing.JButton ButtonPendaftaranObat;
465 private javax.swing.JButton ButtonPendaftaranPasien;
466 private javax.swing.JButton ButtonResepPasien;
467 private javax.swing.JPanel PanelsidebarAdmin2;
468 private javax.swing.JTable TabelResepPasien;
469 private javax.swing.JLabel jLabel1;
470 private javax.swing.JPanel jPanel1;
471 private javax.swing.JScrollPane jScrollPane1;
472 // End of variables declaration
473 }
```

Penjelasan ResepPasienAdmin.java

a. Deskripsi Umum

Antarmuka untuk admin guna mengelola data resep pasien. Admin dapat melihat, menambah, mengubah, atau menghapus resep.

b. Fungsionalitas Utama

- Menampilkan riwayat resep pasien, termasuk penyakit, tanggal, nama pasien, dokter, obat, dan status.
- Menyediakan tombol "Action" untuk melakukan operasi CRUD pada data resep.

c. Komponen Penting (Code Breakdown)

- `tampilkanDataResepPasien()`: Mengambil data resep yang sudah digabungkan (join) dari `ResepPasienDAO.getAllResepLengkap()` dan menampilkannya pada tabel.
- `ButtonActionResepPasienActionPerformed(...)`: Seperti pada `JadwalDokterAdmin`, tombol ini menggunakan pola Controller. Ia memanggil `new ResepPasienController("admin").showActionDialog()`. Controller ini akan mengelola logika untuk menambah, mengedit (misalnya mengubah status resep), atau menghapus resep. Setelah aksi selesai, tabel akan di-refresh.

O. ResepPasienDokter.java

```
5   package klinik_pbo;
6
7   import controller.ResepPasienController;
8   import javax.swing.table.DefaultTableModel;
9   import klinik_pbo.koneksii;
10  import java.sql.Connection;
11  import java.sql.DriverManager;
12  import java.sql.SQLException;
13  import java.util.List;
14  import java.util.logging.Level;
15  import java.util.logging.Logger;
16  import javax.swing.JOptionPane;
17  import javax.swing.SwingUtilities;
18  import model.ResepPasien;
19  import dao.ResepPasienDAO;
20
21  /**
22  *
23  * @author avvjelly
24  */
25  public class ResepPasienDokter extends javax.swing.JFrame {
26      private void tampilkanDataResepPasien() {
27          List<ResepPasien> daftar = ResepPasienDAO.getAllResepLengkap();
28          DefaultTableModel model = (DefaultTableModel) TabelResepPasien.getModel();
29          model.setRowCount(0); // Bersihkan isi tabel
30
31          for (ResepPasien r : daftar) {
32              Object[] row = {
33                  r.getIdResep(),
34                  r.getPenyakit(),
35                  r.getTanggal(),
36                  r.getNamaPasien(),
37                  r.getNamaDokter(),
38                  r.getNamaObat(),
39                  r.getStatus()
34              };
35              model.addRow(row);
36          }
37      }
38  }
```

```
49  public ResepPasienDokter () {
50      initComponents();
51      tampilkanDataResepPasien();
52  }
53
54  /**
55  * This method is called from within the constructor to initialize the form.
56  * WARNING: Do NOT modify this code. The content of this method is always
57  * regenerated by the Form Editor.
58  */
59  @SuppressWarnings("unchecked")
60  // Generated Code
61
62  private void ButtonDashboardActionPerformed(java.awt.event.ActionEvent evt) {
63      // TODO add your handling code here:
64      // 1. Tutup form saat ini
65      this.dispose();
66
67      // 2. Buka form
68      DashboardDokter dashboard = new DashboardDokter();
69      dashboard.setVisible(true);
70  }
71
72  private void ButtonJadwalDokterActionPerformed(java.awt.event.ActionEvent evt) {
73      // TODO add your handling code here:
74      // 1. Tutup form saat ini
75      this.dispose();
76
77      // 2. Buka form
78      JadwalDokterDokter daftarjadwal = new JadwalDokterDokter();
79      daftarjadwal.setVisible(true);
80  }
81
82  private void ButtonResepPasienActionPerformed(java.awt.event.ActionEvent evt) {
83      // TODO add your handling code here:
84      // 1. Tutup form saat ini
85      this.dispose();
86
87      // 2. Buka form
88      ResepPasienDokter daftarresep = new ResepPasienDokter();
89      daftarresep.setVisible(true);
90  }
91
92  
```

```

253     private void ButtonLogoutActionPerformed(java.awt.event.ActionEvent evt) {
254         // TODO add your handling code here:
255         // Tampilkan dialog konfirmasi
256         int pilihan = JOptionPane.showConfirmDialog(this,
257             "Apakah Anda yakin ingin logout?", "Konfirmasi Logout",
258             JOptionPane.YES_NO_OPTION);
259
260         // Jika pengguna memilih "Yes"
261         if (pilihan == JOptionPane.YES_OPTION) {
262             // 1. Tutup form saat ini
263             this.dispose();
264
265             // 2. Buka form login
266             Login login = new Login();
267             login.setVisible(true);
268         }
269     }
270
271     private void ButtonActionResepPasienActionPerformed(java.awt.event.ActionEvent evt) {
272         // TODO add your handling code here:
273         new ResepPasienController("dokter").showActionDialog();
274         tampilkanDataResepPasien(); // refresh tabel jika kamu punya
275     }
276
277
278     /**
279      * @param args the command line arguments
280     */
281     public static void main(String args[]) {
282         /* Set the Nimbus look and feel */
283         [ Look and feel setting code (optional) ]
284
285         /* Create and display the form */
286         java.awt.EventQueue.invokeLater(new Runnable() {
287             public void run() {
288                 new ResepPasienDokter().setVisible(true);
289             }
290         });
291     }
292
293
294
295     /**
296      * Variables declaration - do not modify
297      */
298     private javax.swing.JButton ButtonActionResepPasien;
299     private javax.swing.JButton ButtonDashboard;
300     private javax.swing.JButton ButtonJadwalDokter;
301     private javax.swing.JButton ButtonLogout;
302     private javax.swing.JButton ButtonResepPasien;
303     private javax.swing.JPanel PanelSidebarAdmin2;
304     private javax.swing.JTable TabelResepPasien;
305     private javax.swing.JLabel jLabel1;
306     private javax.swing.JPanel jPanel1;
307     private javax.swing.JScrollPane jScrollPane1;
308
309     // End of variables declaration
310
311 }
312
313
314
315
316
317
318
319
320
321
322
323
324
325 }
```

Penjelasan ResepPasienDokter.java

a. Deskripsi Umum

Halaman ini memungkinkan dokter untuk melihat dan mengelola resep pasien yang mereka tangani.

b. Fungsionalitas Utama

- Menampilkan daftar resep yang telah dibuat.
- Dokter dapat melakukan aksi pada resep, seperti membuat resep baru atau mungkin mengubah status resep yang mereka buat.

c. Komponen Penting (Code Breakdown)

- `tampilkanDataResepPasien()`: Mengambil dan menampilkan semua data resep pasien ke dalam tabel.
- `ButtonActionResepPasienActionPerformed(...)`: Memanggil `new ResepPasienController("dokter").showActionDialog()`. Peran "dokter" yang

dilewatkan ke controller mungkin membatasi aksi yang bisa dilakukan (misalnya, dokter hanya bisa mengedit resep yang ia buat sendiri, berbeda dengan admin yang bisa mengelola semua resep). Ini adalah contoh implementasi kontrol akses berbasis peran yang baik.

P. Koneksi.java

```
1 package klinik_pbo;
2
3 import java.sql.Connection;
4 import java.sql.DriverManager;
5 import java.sql.SQLException;
6 import java.util.logging.Level;
7 import java.util.logging.Logger;
8 import javax.swing.JOptionPane;
9
10 public class koneksi {
11     private static Connection conn;
12
13     // Konfigurasi database (sesuaikan dengan setting database Anda)
14     private static final String JDBC_DRIVER = "com.mysql.cj.jdbc.Driver";
15     private static final String DB_URL = "jdbc:mysql://localhost:3306/klinik_db";
16     private static final String USER = "root";
17     private static final String PASS = "";
18
19     // Method untuk mendapatkan koneksi
20     public static Connection getKoneksi() {
21         try {
22             if (conn == null || conn.isClosed()) {
23                 Class.forName(JDBC_DRIVER);
24                 conn = DriverManager.getConnection(DB_URL, USER, PASS);
25                 System.out.println("Koneksi ke database berhasil!");
26             }
27         } catch (ClassNotFoundException | SQLException ex) {
28             Logger.getLogger(koneksi.class.getName()).log(Level.SEVERE, null, ex);
29             System.out.println("Koneksi ke database gagal: " + ex.getMessage());
30             JOptionPane.showMessageDialog(null, "Gagal terhubung ke database: " + ex.getMessage(),
31                                         "Database Error", JOptionPane.ERROR_MESSAGE);
32         }
33         return conn;
34     }
35
36     // Method untuk menutup koneksi
37     public static void closeConnection() {
38         if (conn != null) {
39             try {
40                 conn.close();
41                 conn = null;
42                 System.out.println("Koneksi database ditutup.");
43             }
44         }
45     }
46
47     // Method untuk menutup koneksi
48     public static void closeConnection() {
49         if (conn != null) {
50             try {
51                 conn.close();
52                 conn = null;
53                 System.out.println("Koneksi database ditutup.");
54             } catch (SQLException ex) {
55                 Logger.getLogger(koneksi.class.getName()).log(Level.SEVERE, null, ex);
56             }
57         }
58     }
59 }
```

Penjelasan Koneksi.java

a. Deskripsi Umum

koneksi.java adalah kelas utilitas yang sangat penting dan fundamental dalam aplikasi ini. Kelas ini bertanggung jawab penuh untuk mengelola koneksi ke database MySQL, menerapkan pola desain Singleton untuk objek koneksi.

b. Fungsionalitas Utama

- Menyediakan satu titik akses terpusat untuk mendapatkan koneksi database (Connection).
- Mengonfigurasi parameter koneksi seperti JDBC driver, URL database, username, dan password.
- Mencegah pembuatan koneksi berulang kali dengan memeriksa apakah koneksi sudah ada dan masih terbuka.
- Menyediakan metode untuk menutup koneksi dengan aman.

c. Komponen Penting (Code Breakdown)

- Variabel Statis:
 - private static Connection conn;: Menyimpan satu-satunya instance koneksi untuk seluruh aplikasi.
 - JDBC_DRIVER, DB_URL, USER, PASS: Konstanta statis yang membuat konfigurasi database mudah diubah di satu tempat.
- getKoneksi(): Metode pabrik statis (static factory method) untuk mendapatkan koneksi.
- Mekanisme if (conn == null || conn.isClosed()) adalah kunci dari pola Singleton di sini. Ini memastikan bahwa koneksi baru hanya akan dibuat jika belum ada atau jika koneksi sebelumnya telah ditutup.
- Blok try-catch menangani SQLException dan ClassNotFoundException jika driver database tidak ditemukan atau koneksi gagal, lalu menampilkan JOptionPane error kepada pengguna.
- closeConnection(): Metode statis untuk menutup koneksi yang sedang aktif dan mengatur ulang variabel conn menjadi null, membebaskan sumber daya database.

Q. Klinik_PBO.java

```
1  /*
2  * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
3  * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Main.java to edit this template
4  */
5  package klinik_pbo;
6
7  /**
8  *
9  * @author avvjelly
10 */
11 public class Klinik_PBO {
12
13     /**
14      * @param args the command line arguments
15     */
16     public static void main(String[] args) {
17         // TODO code application logic here
18     }
19
20 }
21
```

Penjelasan Klinik_PBO.java

a. Deskripsi Umum

Klinik_PBO.java adalah kelas utama yang secara teknis dihasilkan oleh IDE saat proyek dibuat. Kelas ini berisi metode main standar yang merupakan titik masuk (entry point) untuk aplikasi Java.

b. Fungsionalitas Utama

Sebagai titik awal eksekusi program oleh Java Virtual Machine (JVM).

c. Komponen Penting (Code Breakdown)

main(String[] args): Metode utama aplikasi. Dalam file ini, metode main dibiarkan kosong (// TODO code application logic here). Ini adalah praktik umum dalam pengembangan aplikasi Swing dengan IDE seperti NetBeans. Meskipun ini adalah kelas utama secara teknis, eksekusi aplikasi sebenarnya dimulai dari file GUI lain (dalam kasus ini, kemungkinan besar Login.java) yang telah ditetapkan sebagai "Main Class" di properti proyek. IDE akan secara otomatis memanggil metode main dari kelas Login.java saat proyek dijalankan, yang kemudian akan membuat dan menampilkan jendela login.

3.2.4 Folder Model

Bagian ini menguraikan kelas-kelas dalam paket model. Kelas-kelas dalam paket ini adalah representasi dari entitas-entitas dunia nyata yang ada dalam sistem klinik. Mereka sering disebut sebagai POJO (Plain Old Java Object) dan hanya berisi *fields* (atribut) untuk menyimpan data, serta *constructor* untuk inisialisasi dan metode *getter/setter* untuk mengakses dan memodifikasi data.

A. User.java

```
5 package model;
6
7 public abstract class User {
8     protected String idUser;
9     protected String nama;
10
11     public User(String idUser, String nama) {
12         this.idUser = idUser;
13         this.nama = nama;
14     }
15
16     public String getIdUser() {
17         return idUser;
18     }
19
20     public String getNama() {
21         return nama;
22     }
23
24     public abstract String getRole();
25 }
```

Penjelasan User.java

2. Deskripsi Umum

User.java adalah kelas dasar (base class) untuk semua jenis pengguna dalam sistem. Kelas ini mendefinisikan atribut-atribut umum yang dimiliki oleh setiap pengguna, seperti informasi login.

3. Fungsionalitas Utama

- Menyimpan informasi dasar pengguna: idUser, nama, username, password, dan role.
- Menyediakan dua *constructor*: satu minimal (hanya idUser dan nama) dan satu lengkap.
- Menyediakan metode *getter* dan *setter* untuk semua atribut.

4. Komponen Penting (Code Breakdown)

- private String role;: Atribut untuk menentukan peran pengguna (misalnya, "admin", "dokter").
- getRole(): Metode ini memiliki logika tambahan untuk memastikan bahwa jika peran tidak diatur, nilai default "user" akan dikembalikan.

B. Admin.java

```
package model;

public class Admin extends User {
    private String idAdmin;

    public Admin(String idAdmin, String nama, String idUser) {
        super(idUser, nama);
        this.idAdmin = idAdmin;
    }

    public String getIdAdmin() {
        return idAdmin;
    }

    @Override
    public String getRole() {
        return "admin";
    }
}
```

Penjelasan Admin.java

1. Deskripsi Umum

Admin.java adalah kelas yang merepresentasikan pengguna dengan hak akses admin. Kelas ini merupakan turunan (subclass) dari kelas User.

2. Fungsionalitas Utama

- Mewarisi semua atribut dan metode dari kelas User.
- Menambahkan atribut spesifik admin, yaitu idAdmin.
- Meng-override metode getRole() untuk selalu mengembalikan nilai "admin".

3. Komponen Penting (Code Breakdown)

- `public class Admin extends User:` Menandakan bahwa Admin adalah turunan dari User.
- `super(idUser, nama):` Memanggil *constructor* dari kelas induk (User) untuk menginisialisasi atribut idUser dan nama.
- `@Override public String getRole():` Mengganti implementasi metode getRole() dari kelas induk untuk memastikan setiap objek Admin memiliki peran "admin".

C. Dokter.java

```
5 package model;
6
7 public class Dokter extends User {
8     private String idDokter;
9     private String spesialisasi;
10
11     public Dokter(String idDokter, String nama, String spesialisasi, String idUser) {
12         super(idUser, nama);
13         this.idDokter = idDokter;
14         this.spesialisasi = spesialisasi;
15     }
16
17     public String getIdDokter() {
18         return idDokter;
19     }
20
21     public String getSpesialisasi() {
22         return spesialisasi;
23     }
24
25     @Override
26     public String getRole() {
27         return "dokter";
28     }
29 }
```

Penjelasan Dokter.java

d. Deskripsi Umum

Dokter.java merepresentasikan entitas dokter dalam sistem dan merupakan turunan dari kelas User.

e. Fungsionalitas Utama

- Mewarisi atribut dan metode dari User.
- Menambahkan atribut spesifik dokter: idDokter dan spesialisasi.
- Meng-override metode getRole() untuk selalu mengembalikan nilai "dokter".

f. Komponen Penting (Code Breakdown)

- `public class Dokter extends User`: Menandakan pewarisan dari kelas User.
- `super(idUser, nama)`: Memanggil *constructor* dari kelas User.
- `@Override public String getRole()`: Memastikan setiap objek Dokter memiliki peran "dokter".

D. Pasien.java

```
5 package model;
6
7 public class Pasien {
8     private String idPasien;
9     private String nama;
10    private String alamat;
11    private String telepon;
12
13    public Pasien(String idPasien, String nama, String alamat, String telepon) {
14        this.idPasien = idPasien;
15        this.nama = nama;
16        this.alamat = alamat;
17        this.telepon = telepon;
18    }
19
20    // Getter
21    public String getIdPasien() {
22        return idPasien;
23    }
24
25    public String getNama() {
26        return nama;
27    }
28
29    public String getAlamat() {
30        return alamat;
31    }
32
33    public String getTelepon() {
34        return telepon;
35    }
36
37    // Setter (jika kamu butuh ubah datanya)
38    public void setIdPasien(String idPasien) {
39        this.idPasien = idPasien;
40    }
41
42    public void setNama(String nama) {
43        this.nama = nama;
44    }
45
46    public void setAlamat(String alamat) {
47        this.alamat = alamat;
48
49
50    public void setTelepon(String telepon) {
51        this.telepon = telepon;
52    }
53}
```

Penjelasan Pasien.java

a. Deskripsi Umum

Pasien.java adalah kelas model yang merepresentasikan data seorang pasien.

b. Fungsionalitas Utama

- Menyimpan atribut-atribut pasien: idPasien, nama, alamat, dan telepon.
- Menyediakan satu *constructor* untuk menginisialisasi semua atribut.
- Menyediakan metode *getter* dan *setter* untuk setiap atribut.

c. Komponen Penting (Code Breakdown)

- `private String idPasien;` : Atribut untuk menyimpan ID unik pasien.
- `private String nama;` : Atribut untuk menyimpan nama pasien.
- `private String alamat;` : Atribut untuk menyimpan alamat pasien.
- `private String telepon;` : Atribut untuk menyimpan nomor telepon pasien.
- `Pasien(...)` : Konstruktor untuk menginisialisasi data pasien saat objek dibuat.

- getIdPasien() sampai getTelepon() : Method getter untuk mengambil nilai masing-masing atribut.
- setIdPasien(...) sampai setTelepon(...) : Method setter untuk mengubah nilai atribut jika dibutuhkan.

E. Obat.java

```

5   package model;
6
7   /**
8   * 
9   * @author avvjelly
10  */
11 public class Obat {
12     private String idObat;
13     private String namaObat;
14     private int stok;
15     private double harga;
16
17     public Obat(String idObat, String namaObat, int stok, double harga) {
18         this.idObat = idObat;
19         this.namaObat = namaObat;
20         this.stok = stok;
21         this.harga = harga;
22     }
23
24     // Getters
25     public String getIdObat() {
26         return idObat;
27     }
28
29     public String getNamaObat() {
30         return namaObat;
31     }
32
33     public int getStok() {
34         return stok;
35     }
36
37     public double getHarga() {
38         return harga;
39     }
40
41     // Setters
42     public void setIdObat(String idObat) {
43         this.idObat = idObat;
44     }
45
46     public void setNamaObat(String namaObat) {
47         this.namaObat = namaObat;
48     }
49
50     public void setStok(int stok) {
51         this.stok = stok;
52     }
53
54     public void setHarga(double harga) {
55         this.harga = harga;
56     }
57 }
```

Penjelasan Obat.java

- Deskripsi Umum Obat.java adalah kelas model untuk merepresentasikan entitas obat dalam inventaris klinik.
- Fungsionalitas Utama

- Menyimpan data obat: idObat, namaObat, stok, dan harga.
- Menyediakan *constructor* dan metode *getter/setter* lengkap untuk semua atributnya.

c. Komponen Penting (Code Breakdown)

- private String idObat; : Atribut untuk menyimpan ID unik obat.
- private String namaObat; : Atribut untuk menyimpan nama obat.
- private int stok; : Atribut untuk menyimpan jumlah stok obat.
- private double harga; : Atribut untuk menyimpan harga obat.
- Obat(...) : Konstruktor untuk menginisialisasi data obat saat objek dibuat.
- getIdObat() sampai getHarga() : Method getter untuk mengambil nilai masing-masing atribut.
- setIdObat(...) sampai setHarga(...) : Method setter untuk mengubah nilai atribut jika dibutuhkan.

F. Ruangan.java

```

5   package model;
6
7   /**
8   * 
9   * @author avvjelly
10  */
11 public class Ruangan {
12     private String idRuangan;
13     private String namaRuangan;
14     private int lantai;
15     private int kapasitas;
16
17     public Ruangan(String idRuangan, String namaRuangan, int lantai, int kapasitas) {
18         this.idRuangan = idRuangan;
19         this.namaRuangan = namaRuangan;
20         this.lantai = lantai;
21         this.kapasitas = kapasitas;
22     }
23
24     // Getters
25     public String getIdRuangan() {
26         return idRuangan;
27     }
28
29     public String getNamaRuangan() {
30         return namaRuangan;
31     }
32
33     public int getLantai() {
34         return lantai;
35     }
36
37     public int getKapasitas() {
38         return kapasitas;
39     }
40
41     // Setters
42     public void setIdRuangan(String idRuangan) {
43         this.idRuangan = idRuangan;
44     }
45
46     public void setNamaRuangan(String namaRuangan) {
47         this.namaRuangan = namaRuangan;
48     }
49
50 }
```

```
41     // Setters
42     public void setIdRuangan(String idRuangan) {
43         this.idRuangan = idRuangan;
44     }
45
46     public void setNamaRuangan(String namaRuangan) {
47         this.namaRuangan = namaRuangan;
48     }
49
50     public void setLantai(int lantai) {
51         this.lantai = lantai;
52     }
53
54     public void setKapasitas(int kapasitas) {
55         this.kapasitas = kapasitas;
56     }
57
58 }
59
```

Penjelasan Ruangan.java

a. Deskripsi Umum

Ruangan.java merepresentasikan entitas ruangan fisik yang ada di klinik.

b. Fungsionalitas Utama

- Menyimpan data ruangan: idRuangan, namaRuangan, lantai, dan kapasitas.
- Menyediakan *constructor* dan metode *getter/setter* standar.

c. Komponen Penting (Code Breakdown)

- **public class Ruangan:** Membuat class Ruangan untuk merepresentasikan data ruangan.
- **private String idRuangan, namaRuangan; private int lantai, kapasitas;:** Variabel untuk menyimpan informasi ruangan (ID, nama, lantai, dan kapasitas).
- **public Ruangan(...):** Constructor untuk mengisi data saat objek dibuat.
- **this.nama = nama;:** Mengisi nilai atribut dengan parameter dari constructor.
- **getIdRuangan(), getNamaRuangan(), getLantai(), getKapasitas():** Method getter untuk mengambil nilai masing-masing atribut.
- **setIdRuangan(...), setNamaRuangan(...), setLantai(...), setKapasitas(...):** Method setter untuk mengubah nilai atribut.

G. JadwalPraktik.java

```
5     package model;
6
7     /**
8      *
9      * @author avvjelly
10     */
11    public class JadwalPraktik {
12        private String idJadwal;
13        private String hari;
14        private String jamMulai;
15        private String jamBerakhir;
16
17        public JadwalPraktik(String idJadwal, String hari, String jamMulai, String jamBerakhir) {
18            this.idJadwal = idJadwal;
19            this.hari = hari;
20            this.jamMulai = jamMulai;
21            this.jamBerakhir = jamBerakhir;
22        }
23
24        public String getIdJadwal() { return idJadwal; }
25        public String getHari() { return hari; }
26        public String getJamMulai() { return jamMulai; }
27        public String getJamBerakhir() { return jamBerakhir; }
28    }
29
```

Penjelasan JadwalPraktik.java

a. Deskripsi Umum

JadwalPraktik.java adalah kelas model yang digunakan untuk menampung informasi jadwal praktik secara umum.

b. Fungsionalitas Utama

- Menyimpan atribut jadwal: idJadwal, hari, jamMulai, dan jamBerakhir.
- Hanya menyediakan metode *getter*, menunjukkan bahwa kelas ini dirancang untuk menjadi objek transfer data yang *read-only* setelah dibuat.

c. Komponen Penting (Code Breakdown)

- idJadwal: ID unik untuk jadwal praktik.
- hari: Menyimpan hari praktik.
- jamMulai: Jam mulai praktik.
- jamBerakhir: Jam berakhir praktik.
- Konstruktor: Menginisialisasi semua atribut saat objek dibuat.
- Getter (getIdJadwal, getHari, getJamMulai, getJamBerakhir): Mengambil nilai dari masing-masing atribut.

H. JadwalDokter.java

```
5 package model;
6
7 public class JadwalDokter {
8     private String idJadwalDokter;
9     private String namaDokter;
10    private String hari;
11    private String namaRuangan;
12
13    public JadwalDokter(String idJadwalDokter, String namaDokter, String hari, String namaRuangan) {
14        this.idJadwalDokter = idJadwalDokter;
15        this.namaDokter = namaDokter;
16        this.hari = hari;
17        this.namaRuangan = namaRuangan;
18    }
19
20    public String getIdJadwalDokter() {
21        return idJadwalDokter;
22    }
23
24    public String getNamaDokter() {
25        return namaDokter;
26    }
27
28    public String getHari() {
29        return hari;
30    }
31
32    public String getNamaRuangan() {
33        return namaRuangan;
34    }
35
36 }
```

Penjelasan JadwalDokter.java

a. Deskripsi Umum

JadwalDokter.java adalah kelas model yang digunakan untuk menampung data hasil penggabungan (join) dari beberapa tabel terkait jadwal. Ini bukan representasi langsung dari satu tabel, melainkan struktur data untuk menampilkan informasi jadwal yang lebih lengkap.

b. Fungsionalitas Utama

- Menyimpan data gabungan: idJadwalDokter, namaDokter, hari, dan namaRuangan.
- Hanya menyediakan metode *getter*, menandakan perannya sebagai objek transfer data yang tidak dapat diubah setelah dibuat.

c. Komponen Penting (Code Breakdown)

- private String idJadwalDokter;: Menyimpan ID unik dari jadwal praktik dokter.
- private String namaDokter;: Menyimpan nama dokter yang bersangkutan.
- private String hari;: Menyimpan hari praktik dokter.
- private String namaRuangan;: Menyimpan nama ruangan tempat praktik berlangsung.
- public JadwalDokter(...): Konstruktor untuk menginisialisasi semua atribut saat objek dibuat.
- getIdJadwalDokter(), getNamaDokter(), getHari(), getNamaRuangan(): Method getter untuk mengambil nilai atribut masing-masing secara aman sesuai prinsip enkapsulasi.

I. ResepPasien.java

```
1 package model;
2
3 import java.sql.Date;
4
5 public class ResepPasien {
6     private String idResep;
7     private String penyakit;
8     private Date tanggal;
9     private String namaPasien;
10    private String namaDokter;
11    private String namaObat;
12    private String status; // tambahan status
13
14    public ResepPasien(String idResep, String penyakit, Date tanggal, String namaPasien, String namaDokter, String namaObat, String status) {
15        this.idResep = idResep;
16        this.penyakit = penyakit;
17        this.tanggal = tanggal;
18        this.namaPasien = namaPasien;
19        this.namaDokter = namaDokter;
20        this.namaObat = namaObat;
21        this.status = status; // set status
22    }
23
24    public String getIdResep() { return idResep; }
25    public String getPenyakit() { return penyakit; }
26    public Date getTanggal() { return tanggal; }
27    public String getNamaPasien() { return namaPasien; }
28    public String getNamaDokter() { return namaDokter; }
29    public String getNamaObat() { return namaObat; }
30    public String getStatus() { return status; } // getter status
31
32    public void setIdResep(String idResep) { this.idResep = idResep; }
33    public void setPenyakit(String penyakit) { this.penyakit = penyakit; }
34    public void setTanggal(Date tanggal) { this.tanggal = tanggal; }
35    public void setNamaPasien(String namaPasien) { this.namaPasien = namaPasien; }
36    public void setNamaDokter(String namaDokter) { this.namaDokter = namaDokter; }
37    public void setNamaObat(String namaObat) { this.namaObat = namaObat; }
38    public void setStatus(String status) { this.status = status; } // setter status
39
40}
```

Penjelasan ResepPasien.java

a. Deskripsi Umum

ResepPasien.java adalah kelas model yang mirip dengan JadwalDokter.java, yaitu digunakan untuk menampung data hasil JOIN dari beberapa tabel (resep, pasien, dokter, obat).

b. Fungsionalitas Utama

- Menyimpan informasi resep yang komprehensif: idResep, penyakit, tanggal, namaPasien, namaDokter, namaObat, dan status.
- Menyediakan *constructor* serta metode *getter* dan *setter* untuk semua atributnya.

c. Komponen Penting (Code Breakdown)

- idResep, penyakit, tanggal, namaPasien, namaDokter, namaObat, status: Atribut yang merepresentasikan data penting dalam sebuah resep pasien.
- Konstruktor ResepPasien(...): Menginisialisasi semua data resep saat objek dibuat.
- Getter & Setter: Digunakan untuk mengambil dan mengubah nilai atribut secara aman sesuai prinsip enkapsulasi.

BAB IV

KESIMPULAN

Dengan adanya fitur-fitur tersebut, penggunaan program klinik sederhana dapat membantu meningkatkan efektivitas dan efisiensi pelayanan kesehatan. Admin dapat dengan mudah mendaftarkan pasien baru dan mengelola informasi penting seperti data dokter, jadwal praktik, serta ketersediaan obat dan ruangan. Dokter pun terbantu dengan fitur resep obat yang memungkinkan penulisan resep secara sistematis dan terdokumentasi. Selain itu, pasien dapat dilayani dengan lebih cepat karena semua informasi tersedia secara terstruktur dan terpusat. Secara keseluruhan, aplikasi ini mampu mengurangi kesalahan pencatatan manual dan mendukung kelancaran operasional klinik sehari-hari.

Berdasarkan hasil pengembangan dan pembahasan yang telah dilakukan, dapat disimpulkan bahwa program klinik sederhana berbasis GUI yang dikembangkan dengan pendekatan Object-Oriented Programming (OOP) menggunakan bahasa pemrograman Java mampu memberikan solusi efektif untuk mendukung operasional pelayanan klinik. Penggunaan konsep OOP memungkinkan sistem memiliki struktur yang modular, mudah dipelihara, dan dikembangkan di masa depan.

Fitur-fitur utama yang diimplementasikan, seperti pendaftaran pasien, pengelolaan data dokter, jadwal praktik, resep obat, daftar obat, dan pengaturan ruangan, terbukti mampu meningkatkan efektivitas dan efisiensi dalam administrasi klinik. Sistem ini memberikan kemudahan bagi admin dalam mengelola data secara menyeluruh dengan dukungan fungsi CRUD (Create, Read, Update, Delete), serta membantu dokter dalam melakukan praktik dan pemberian resep secara tertib dan terdokumentasi.

Secara keseluruhan, program klinik sederhana ini tidak hanya berfungsi sebagai alat bantu administrasi, namun juga sebagai media pembelajaran yang aplikatif untuk memahami penerapan konsep OOP dalam pengembangan perangkat lunak berbasis GUI. Ke depannya, sistem ini masih dapat dikembangkan lebih lanjut, seperti integrasi dengan database online, sistem antrian otomatis, dan fitur laporan statistik untuk menunjang manajemen klinik secara lebih profesional.